Când serverul citește conținutul tabelului `user`, sortează intrările în funcție de valorile din coloanele `Host` și `User`. Valoarea `Host` este valoare dominantă (intrările cu aceeași valoare `Host` sunt sortate împreună, apoi ordonate în funcție de valoarea din coloana `User`). Totuși, sortarea *nu* se execută în ordine lexicografică, ci numai parțial. Principiul care trebuie reținut este acela că valorile literale sunt preferate în detrimentul modelelor. Aceasta înseamnă că, dacă vă conectați de la `boa.snake.net` și există intrări cu valorile `boa.snake.net` și `%.snake.net` în coloana `Host`, prima intrare va fi preferată. Similar, `%.snake.net` este preferată față de `%.net`, care la rândul său este preferată față de `%`. Stabilirea corespondențelor pentru numerele IP funcționează în același mod. Pentru un client care se conectează de la o gazdă cu un număr IP `192.168.3.14`, vor corespunde intrările `192.168.3.14`, `192.168.3.%`, `192.168.%`, `192.%` și `%`, în această ordine.

Riscuri de evitat în ceea ce privește tabelele de acordare

Această secțiune descrie unele măsuri de precauție care trebuie luate în considerare atunci când acordați privilegii, precum și riscurile care însoțesc opțiunile nechibzuite. În general, este de dorit ca privilegiile de super-utilizator să fie acordate cu economie. Cu alte cuvinte, nu activați privilegiile din intrările tabelului `user`. Utilizați în schimb celelalte tabele de acordare, pentru a limita privilegiile utilizatorilor la anumite baze de date, tabele sau coloane. Privilegiile din tabelul `user` permit utilizatorului să influențeze operarea serverului dumneavoastră, respectiv să obțină accesul la orice tabel din orice bază de date.

Nu acordați privilegii pentru baza de date `mysql`. Un utilizator cu privilegii asupra bazei de date care conține tabelele de acordare poate fi capabil să modifice tabelele pentru a dobândi privilegii și asupra oricărei alte baze de date. Prin acordarea privilegiilor care permit unui utilizator să modifice tabelele din baza de date `mysql`, utilizatorul respectiv dobândește efectiv un privilegiu `GRANT` global. Dacă utilizatorul poate modifica tabelele în mod direct, acest lucru este aproape echivalent cu posibilitatea de a emite orice instrucțiune `GRANT` care îi trece prin cap respectivului.

Privilegiul `FILE` este deosebit de periculos; nu-l acordați cu ușurință. Iată un exemplu de acțiune pe care o poate executa un utilizator cu privilegiul `FILE`:

```
CREATE TABLE etc_parole (pwd_entry TEXT);
LOAD DATA INFILE "/etc/passwd" INTO TABLE etc_parole;
SELECT * FROM etc_parole;
```

După ce a emis aceste instrucțiuni, utilizatorul dispune de conținutul fișierului dumneavoastră cu parole. De fapt, conținutul *oricărui* fișier care poate fi citit public din server este accesibil prin rețea de către orice utilizator cu privilegiul `FILE`.

De asemenea, privilegiul `FILE` poate fi exploatat pentru a compromite bazele de date pe sisteme care nu sunt configurate cu permisiuni de fișier suficient de restrictive. Acesta este un motiv pentru care trebuie să decideți ca numai serverul să poată citi conținutul catalogului de date. Dacă fișierele corespunzătoare tabelelor din bazele de date pot fi citite de oricine, atunci pot fi citite nu numai de către orice utilizator cu un cont pe gazda serverului, dar și de orice utilizator al programelor client cu privilegiul `FILE` care se poate conecta prin rețea! Următoarea procedură demonstrează cum se poate lucra:

- Creați un tabel cu o coloană `LONGBLOB`:

```
USE test
CREATE TABLE tmp (b LONGBLOB)
```

- Folosiți tabelul pentru a citi în acesta conținutul fiecărui fișier corespunzător tabelului pe care doriți să-l furați, după care scrieți conținutul tabelului într-un fișier din propria dumneavoastră bază de date:

```
LOAD DATA INFILE "./other_db/x.frm" INTO TABLE tmp
   FIELDS ESCAPED BY "" LINES TERMINATED BY ""
SELECT * FROM tmp INTO OUTFILE "y.frm"
   FIELDS ESCAPED BY "" LINES TERMINATED BY ""
DELETE FROM tmp
LOAD DATA INFILE "/.other_db/x.ISD" INTO TABLE tmp
   FIELDS ESCAPED BY "" LINES TERMINATED BY ""
SELECT * FROM tmp INTO OUTFILE "y.ISD"
   FIELDS ESCAPED BY "" LINES TERMINATED BY ""
DELETE FROM tmp
LOAD DATA INFILE "/.other_db/x.ISM" INTO TABLE tmp
   FIELDS ESCAPED BY "" LINES TERMINATED BY ""
SELECT * FROM tmp INTO OUTFILE "y.ISM"
   FIELDS ESCAPED BY "" LINES TERMINATED BY ""
```

- În acest moment, aveți un nou tabel `y`, care include conținutul tabelului `other_db.x`, iar dumneavoastră aveți acces total la acesta.

Pentru a evita să fiți atacat în acest mod, stabiliți permisiunile asupra conținutului catalogului dumneavoastră de date în conformitate cu instrucțiunile din secțiunea „Securitate internă: securizarea accesului la catalogul de date". De asemenea, puteți folosi opțiunea `--skip-show-database` atunci când porniți serverul, pentru a limita folosirea instrucțiunilor `SHOW DATABASES` și `SHOW TABLES` de către utilizatori pentru baze de date la care aceștia nu au acces. Astfel, utilizatorii nu mai pot afla despre existența unor baze de date, respectiv tabele, la care nu trebuie să aibă acces.

Privilegiul `ALTER` poate fi folosit în moduri pe care nu le imaginați. Să presupunem că dorim ca utilizatorul `user1` să poată obține accesul la `tabel1`, dar nu și la `tabel2`. Un utilizator cu privilegiul `ALTER` vă poate deturna intenția folosind `ALTER TABLE` pentru a redenumi `tabel2` cu numele `tabel1`.

Fiți atent la privilegiul `GRANT`. Doi utilizatori cu privilegii diferite, care dispun ambii de privilegiul `GRANT`, își pot mări reciproc puterea drepturilor de acces ale fiecăruia.

Configurarea utilizatorilor fără instrucțiunea GRANT

Dacă dispuneți de o versiune a sistemului MySQL anterioară versiunii 3.22.11, nu puteți folosi instrucțiunile `GRANT` sau `REVOKE` pentru a configura utilizatori și privilegii de acces, dar puteți aduce modificări în mod direct conținutului tabelelor de acordare. O bună parte din aceste operații sunt mai simple dacă înțelegeți cum acționează instrucțiunea `GRANT` pentru a modifica tabelele de acordare. Apoi, puteți proceda în același mod prin

emiterea manuală a instrucțiunilor INSERT. (Instrucțiunea INSERT poate fi dizgrațioasă și dificil de introdus corect, dar aceasta este o altă problemă. „Urâțenia" instrucțiunii INSERT este, de fapt, unul dintre motivele pentru care instrucțiunea GRANT este cu mult mai simplu de utilizat.)

Când emiteți o instrucțiune GRANT, specificați un nume de utilizator și un nume de gazdă, eventual o parolă. Pentru utilizator este creată o intrare în tabelul user, iar aceste valori sunt înregistrate în coloanele User, Host și Password ale intrării. Dacă specificați privilegii globale în instrucțiunea GRANT, aceste privilegii sunt înregistrate în coloanele cu privilegii ale intrării. Trebuie să fiți atent la faptul că instrucțiunea GRANT criptează parola automat, dar INSERT nu execută această operație; va trebui să folosiți funcția PASSWORD() pentru a cripta parolele în instrucțiunile dumneavoastră INSERT.

Dacă ați specificat privilegii la nivel de bază de date, numele de utilizator și numele gazdei sunt înregistrate în coloanele User și Host ale unei intrări din tabelul db. Baza de date pentru care ați acordat privilegii este înregistrată în coloana Db, iar privilegiile pe care le-ați acordat sunt înregistrate în coloanele pentru privilegii.

Pentru privilegiile la nivel de tabel, respectiv la nivel de coloană, efectele sunt asemănătoare. Sunt create intrări în tabelele tables_priv și columns_priv pentru a înregistra numele de utilizator, numele gazdei și baza de date, precum și tabelul, respectiv tabelul și coloana, dacă este necesar. Privilegiile acordate sunt înregistrate în coloana cu privilegii.

Dacă ați reținut descrierea anterioară, trebuie să puteți executa operațiile efectuate de instrucțiunea GRANT chiar în absența acesteia. Rețineți că, atunci când modificați în mod direct tabelele de acordare, va trebui să indicați serverului să le reîncarce, deoarece în caz contrar acesta nu va sesiza modificările. Puteți forța o reîncărcare executând o comandă mysqladmin flush-privileges sau mysqladmin reload. Dacă omiteți acest lucru, vă veți întreba de ce serverul nu se comportă așa cum doriți.

Următoarea instrucțiune GRANT creează un super-utilizator dotat cu toate privilegiile, inclusiv capacitatea de a acorda privilegii altor utilizatori:

```
GRANT ALL ON *.* TO elena@localhost IDENTIFIED BY "cafea"
   WITH GRANT OPTION
```

Instrucțiunea va crea o intrare pentru elena@localhost în tabelul user și va activa toate privilegiile din tabelul respectiv, deoarece acolo sunt stocate privilegiile de super-utilizator (globale). Pentru a face același lucru cu INSERT, instrucțiunea este:

```
INSERT INTO user VALUES ("localhost","elena",PASSWORD("cafea")),
   "Y","Y","Y","Y","Y","Y","Y","Y","Y","Y","Y","Y","Y","Y")
```

Asta este o instrucțiune INSERT urâtă de-a binelea! Puteți chiar descoperi că nu funcționează, în funcție de versiunea dumneavoastră a sistemului MySQL. Poate că structura tabelelor de acordare s-a modificat cu o anumită ocazie și s-ar putea să nu aveți 14 coloane cu privilegii în tabelul dumneavoastră user. Folosiți SHOW COLUMNS pentru a afla care sunt coloanele cu privilegii pe care le conține fiecare tabel de acordare și modificați în consecință instrucțiunile dumneavoastră INSERT.

Următoarea instrucțiune GRANT mai creează un utilizator cu statut de super-utilizator, dar numai pentru un singur privilegiu:

```
GRANT RELOAD ON *.* flush@localhost IDENTIFIED BY "flushpass"
```


Poate vă reamintiți de această instrucțiune din capitolul 11, când am creat utilizatorul flush. Instrucțiunea INSERT pentru acest exemplu este ceva mai simplă decât pentru exemplul anterior; este mai simplu să se menționeze numele coloanelor și să se specifice singura coloană cu privilegii. Toate celelalte coloane vor primi în mod prestabilit valoarea "N":

```
INSERT INTO user (Host,User,Password,Reload_priv)
   VALUES("localhost","flush",PASSWORD("flushpass"),"Y")
```

Privilegiile la nivel de bază de date sunt acordate folosind o clauză ON nume_baza_de_date.*, în loc de ON *.*:

```
GRANT ALL ON samp_db.* TO boris@localhost IDENTIFIED BY "rubin"
```

Aceste privilegii nu sunt globale, deci nu vor fi stocate în tabelul user. Trebuie să creăm o intrare în tabelul user (astfel încât utilizatorul să se poată conecta), dar trebuie să mai creăm o intrare și în tabelul db, pentru a înregistra privilegiile la nivel de bază de date:

```
INSERT INTO user (Host,User,Password)
   VALUES("localhost","boris",PASSWORD("rubin"))
INSERT INTO db VALUES("localhost","samp_db","boris",
   "Y","Y","Y","Y","Y","Y","N","Y","Y","Y")
```

Coloana cu valoarea "N" este cea aferentă privilegiului GRANT; pentru o instrucțiune GRANT la nivel de bază de date însoțită de opțiunea WITH GRANT OPTION, valoarea din acea coloană ar fi fost "Y".

Pentru a configura privilegii la nivel de tabel sau la nivel de coloană, folosiți instrucțiuni INSERT pentru tabelele tables_priv sau columns_priv. Desigur, dacă nu dispuneți de instrucțiunea GRANT, nu veți beneficia nici de aceste tabele, deoarece toate au apărut în MySQL în același timp. Dacă aveți tabelele și doriți să le manipulați manual dintr-un anumit motiv, rețineți că nu se activează privilegiile folosind coloane individuale. Pentru aceasta configurați fie coloana tables_priv.Table_priv, fie coloana columns_priv.Column_priv la o valoare SET alcătuită din privilegiile pe care doriți să le activați. De exemplu, pentru a activa privilegiile INSERT și SELECT pentru un tabel, veți atribui în coloana Table_priv o valoare "Select,Insert" în intrarea relevantă din tabelul tables_priv.

Dacă doriți să modificați privilegiile pentru un utilizator al cărui cont MySQL există deja, folosiți UPDATE în loc de INSERT. Acest fapt este valabil indiferent dacă adăugați sau revocați privilegii. Pentru a elimina complet un utilizator, folosiți DELETE pentru a elimina intrările aferente utilizatorului din fiecare tabel de acordare.

Dacă preferați să nu emiteți interogări pentru a modifica tabelele de acordare în mod direct, puteți examina scripturile mysqlaccess și mysql_setpermissions care sunt incluse în distribuția MySQL.

O enigmă cu privilegii (Partea a II-a)

În capitolul 11, în secțiunea „O enigmă cu privilegii (Partea I)", am examinat o situație de acordare a privilegiilor în care privilegiile nu aveau efectul scontat. Pentru a ne reaminti rapid contextul respectiv, o problemă care se produce frecvent cu instalările noi ale sistemului MySQL este aceea că administratorul dorește să adauge o intrare pentru un utilizator, intrare care să-i permită acestuia să se conecteze de la mai multe gazde. Modalitatea evidentă de a proceda constă în utilizarea unui specificator al numelui gazdei care conține caracterul de înlocuire `%`, astfel încât administratorul creează un utilizator folosind o instrucțiune care seamănă cu aceasta:

```
GRANT ALL ON samp_db.* TO fred@%.snake.net IDENTIFIED BY "cacao"
```

Se întâmplă ca `fred` să aibă un cont pe gazda serverului, deci el încearcă să se conecteze de acolo:

```
% mysql -u fred -pcacao samp_db
ERROR 1045: Access denied for user: 'fred@localhost'
(Using password: YES)
```

De ce? Pentru a înțelege ce se întâmplă, trebuie să examinăm atât modul în care `mysql_install_db` configurează tabelele de acordare inițiale, cât și modul în care serverul folosește intrările din tabelul `user` pentru a stabili echivalențe cu parametrii conexiunilor client. Când vă inițializați bazele de date prin rularea programului `mysql_install_db`, acesta creează intrări în tabelul `user` cu valori în coloana `User` și `Host` astfel:

Host	User
localhost	root
pit-viper.snake.net	root
localhost	
pit-viper.snake.netroot	

Primele două intrări permit utilizatorului `root` să se conecteze la gazda locală specificând fie `localhost`, fie numele gazdei. Celelalte două permit stabilirea unor conexiuni anonime de la gazda locală. La adăugarea intrării pentru utilizatorul `fred`, tabelul va conține următoarele intrări:

Host	User
localhost	root
pit-viper.snake.net	root
localhost	
pit-viper.snake.netroot	
%.snake.netroot	fred

Când serverul pornește, citește intrările și le sortează (mai întâi în funcție de gazdă, apoi în funcție de utilizatorul gazdei), începând de la valorile mai specifice și terminând cu cele mai generale:

Host	User
localhost	root
localhost	
pit-viper.snake.net	root
pit-viper.snake.net	
%.snake.netroot	fred

Cele două intrări cu `localhost` sunt sortate alături, iar apoi intrarea pentru `root` este inserată prima, deoarece este mai specifică decât valoarea vidă. Intrările cu `pit-viper.snake.net` sunt sortate alături într-un mod similar. Toate aceste intrări au valori literale în coloana `Host`, fără caractere de înlocuire, deci toate vor fi sortate anterior intrării pentru utilizatorul `fred`, care folosește un caracter de înlocuire. În particular, intrările pentru utilizatori anonimi au prioritate față de intrarea utilizatorului `fred` în ordinea de sortare.

Rezultatul este acela că, atunci când `fred` încearcă să se conecteze de la gazda locală, un dintre intrările cu nume de utilizator vid este echivalentă cu parametrii de conexiune anterior intrării care conține valoarea `%.snake.net` în coloana `Host`. Parola aferentă intrării respective este vidă, deoarece utilizatorii anonimi prestabiliți nu au parolă. Deoarece `fred` a specificat o parolă atunci când s-a conectat, a apărut o neconcordanță între parole și conexiunea a eșuat.

Aici trebuie reținut că, deși este foarte convenabilă utilizarea caracterelor de înlocuire pentru a specifica gazdele de la care se pot conecta utilizatorii, puteți avea probleme la conexiunile de la gazda locală, atâta timp cât lăsați în tabelul `user` intrările aferente utilizatorilor anonimi.

În general, vă recomand să ștergeți intrările pentru utilizatorii anonimi. Veți avea viață mai ușoară:

```
mysql DELETE FROM user WHERE User="";
```

Dacă doriți să fiți mai concret, ștergeți toate intrările aferente utilizatorilor anonimi din celelalte tabele de acordare. Tabelele care conțin coloane `user` sunt `db`, `tables_priv` și `columns_priv`.

Enigma prezentată în această secțiune a abordat o anumită situație, dar conține o lecție cu un caracter mai general. Dacă privilegiile aferente unui utilizator nu funcționează a cum este de dorit, examinați tabelele de acordare, pentru a vedea dacă există vreo intrare care să conțină în coloana `Host` valori care au un caracter mai specific decât intrarea aferentă utilizatorului în chestiune și care vor corespunde parametrilor conexiunii solicitate de acel utilizator. Dacă aceasta este situația, ați găsit explicația problemei. Poate că este necesar să fiți mai concret în ceea ce privește intrarea aferentă utilizatorului (sau să adăugați o altă intrare, pentru a acoperi situația cu un caracter mai particular).

CAPITOLUL 13

Întreținerea și repararea bazelor de date

Într-un model ideal, MySQL rulează fără probleme începând cu momentul în care îl instalați. Dar uneori se produc și probleme, dintr-o varietate de motive, de la căderi de tensiune și defecțiuni ale componentelor hardware și până la o închidere în condiții inadecvate a serverului MySQL (cum ar fi oprirea execuției acestuia folosind comanda `kill -9` sau când calculatorul suferă o cădere). Circumstanțe precum acestea, din care multe depășesc posibilitatea dumneavoastră de control, pot avea ca rezultat deteriorări ale tabelelor din bazele de date, cauzate în mod caracteristic de scrieri incomplete care se produc în cursul modificării unui tabel.

Acest capitol se concentrează asupra detectării și rezolvării problemelor care pot apărea la tabelele dumneavoastră, indiferent de modul în care se produc aceste probleme. Pentru verificarea și repararea tabelelor, cei mai buni prieteni ai administratorului MySQL sunt utilitarele `myisamchk` și `isamchk`. Aceste două programe dispun de numeroase caracteristici. Deja am discutat despre modul de utilizare a acestora pentru efectuarea analizei distribuției cheilor de index, precum și pentru dezactivarea și reactivarea indexului în capitolul 4, „Optimizarea interogărilor". De asemenea, le puteți folosi pentru verificarea tabelelor și repararea acelora care prezintă probleme, fapt care vă permite să remediați pagubele înainte ca acestea să se agraveze, ducând chiar la imposibilitatea de utilizare a tabelelor.

Pentru o listă completă a opțiunilor acceptate de `myisamchk` și `isamchk`, consultați Anexa E, „Referință de programe MySQL". Pentru informații suplimentare, consultați capitolul „Întreținerea unei instalări a sistemului MySQL" din manualul de referință MySQL.

Verificarea și remedierea tabelelor din bazele de date

Procedura generală pentru detectarea pagubelor aduse unui tabel și pentru remedierea acestora este următoarea:

1. Verificați tabelul pentru a detecta eventualele erori.

   Dacă tabelul nu prezintă erori, ați terminat. În caz contrar, trebuie să-l reparați.
2. Executați copii ale fișierelor din tabel înainte de a începe reparațiile, pentru orice eventualitate.
3. Încercați să reparați tabelul.
4. Dacă operația de reparare eșuează, restaurați tabelul pornind de la copiile de siguranță ale bazei de date și de la jurnalele de actualizare.

Etapa finală a procedurii prezentate anterior presupune că ați efectuat copii de siguranță ale bazei de date și că ați activat caracteristica de consemnare în jurnalele de actualizare. Dacă nu este așa, atunci duceți o existență periculoasă. Citiți capitolul 11, „Administrarea generală a sistemului MySQL", pentru a afla modul în care puteți folosi `mysqldump` și modul de activare a consemnării în jurnal. Nu doriți să vă aflați vreodată în situația de a pierde un tabel fără posibilitatea de a-l recupera, fiindcă v-a fost lene să executați copii de siguranță.

Înainte de a folosi `myisamchk` sau `isamchk` pentru a verifica sau repara tabele, este necesară satisfacerea unor cerințe preliminarii:

- Instituiți proceduri de executare sistematică a copiilor de siguranță a bazelor de date și activați consemnarea în jurnalele de actualizare, în cazul în care răul devine și mai rău și un tabel este atât de deteriorat, încât nu mai poate fi reparat. N-am mai spus asta?
- Parcurgeți în totalitate această secțiune înainte de a încerca vreuna dintre procedurile descrise. În particular, *nu* trebuie să folosiți instrucțiunile prezentate aici înainte de a fi citit secțiunea „Evitarea interacțiunilor cu serverul MySQL". Secțiunea respectivă tratează despre problemele care apar în cazul în care serverul folosește un tabel în același timp în care dumneavoastră încercați să-i aplicați tabelului proceduri de verificare sau reparare. De asemenea, secțiunea descrie modul de a preveni apariția acestor probleme prin păstrarea serverului în stare de repaus în timp ce lucrați.
- Când efectuați verificarea sau repararea tabelelor, trebuie să fiți conectat sub contul folosit pentru a rula `mysqld`, deoarece aveți nevoie de acces de tip citire și scriere la fișierele tabel.

Sintaxa de invocare a utilitarelor `myisamchk` și `isamchk`

Utilitarele MySQL `myisamchk` și `isamchk` sunt foarte asemănătoare și se utilizează în același mod în majoritatea cazurilor. Principala diferență dintre acestea constă în tipul de tabele pentru care este folosit fiecare utilitar. Pentru tabelele MyISAM se folosește `myisamchk`, iar pentru tabelele ISAM se folosește `isamchk`. Se poate determina tipul formatului de stocare folosit de fiecare tabel în funcție de extensia numelui de fișier al fișierului index al tabelului. Extensia `".MYI"` indică un tabel MyISAM, în timp ce extensia `".ISM"` indică un tabel ISAM.

Pentru a folosi oricare din aceste utilitare, indicați-i acestuia tabelele pe care doriți să le verificați sau să le reparați, alături de opțiunile care specifică tipul operației care va fi efectuată:

```
% myisamchk optiuni nume_tabel ...
% isamchk optiuni nume_tabel ...
```

Un argument `nume_tabel` poate fi numele unui tabel sau numele fișierului index al unui tabel. Dacă specificați mai multe tabele, puteți folosi un model de nume de fișier pentru a selecta toate fișierele relevante dintr-un catalog:

```
% myisamchk optiuni *.MYI
% isamchk optiuni *.ISM
```

Nu veți distruge un tabel indicând verificarea acestuia de către un alt program decât cel adecvat, dar programul nu va face nimic altceva decât să emită un mesaj de avertisment. De exemplu, prima din următoarele instrucțiuni va verifica toate tabelele MyISAM din catalogul curent, în timp ce a doua nu va face decât să afișeze mesaje:

```
% myisamchk *.MYI        Corect
% isamchk *.MYI          Greșit; tip de fișier incorect
```

Nici unul dintre cele două utilitare nu face presupuneri cu privire la amplasarea tabelelor, deci trebuie fie să rulați programele în catalogul care conține fișierele tabel, fie să specificați calea de acces spre tabele. Aceasta vă permite să copiați fișierele tabel într-un alt catalog și să utilizați copiile.

Verificarea tabelelor

Atât `myisamchk` cât și `isamchk` furnizează metode de verificare a tabelelor care variază din punct de vedere al profunzimii cu care este examinat tabelul. De regulă, metoda normală este suficientă. Dacă verificarea normală nu raportează erori, dar dumneavoastră suspectați existența unor pagube (poate deoarece interogările par să nu funcționeze în mod adecvat), puteți dori să efectuați o verificare mai aprofundată. Pentru a executa o verificare normală a unui tabel cu oricare din cele două utilitare, invocați-le fără nici o opțiune:

```
% myisamchk nume_tabel
% isamchk nume_tabel
```

Pentru a se executa o verificare amănunțită, folosiți opțiunea `--extend-check`. Această opțiune poate fi foarte lentă, dar execută o verificare extrem de atentă; pentru fiecare înregistrare din fișierul de date al tabelului, este verificată cheia asociată pentru fiecare index din fișierul index, pentru a exista siguranța că indică într-adevăr spre înregistrarea adecvată. `myisamchk` mai are o opțiune intermediară, și anume `--medium-check`, care execută o verificare mai puțin aprofundată decât verificarea amănunțită, dar mai rapidă decât aceasta.

Dacă la o verificare cu opțiunea `--extend-check` nu sunt semnalate erori, puteți fi sigur că tabelul dumneavoastră nu are nici o problemă. Dacă tot mai aveți probleme cu tabelul, cauza trebuie să fie în altă parte. Reexaminați toate interogările care par să determine rezultate problematice, pentru a verifica dacă sunt scrise corect. Dacă sunteți de părere că problema poate veni din partea serverului MySQL, completați un raport privind hibele sau treceți la o versiune mai recentă.

Dacă `myisamchk` sau `isamchk` raportează existența unor erori în tabel, trebuie să le reparați folosind instrucțiunile din secțiunea următoare.

Repararea tabelelor

Repararea tabelelor este o treabă urâtă, mai ales datorită faptului că detaliile au tendința de a fi foarte specifice fiecărui incident în parte. Cu toate acestea, există îndrumări și proceduri de ordin general pe care le puteți respecta pentru a vă mări în mod semnificativ șansele de a putea repara tabelele.

În general, puteți începe cu metoda cea mai rapidă de reparare, pentru a vedea dacă astfel puteți remedia pagubele. Dacă descoperiți că această metodă nu este suficientă, puteți ajunge până la metode mai aprofundate (dar mai lente) până când paguba a fost remediată sau până când nu mai puteți folosi metode și mai aprofundate. (În practică, majoritatea problemelor pot fi remediate fără a se trece la moduri de reparare mai aprofundate și mai lente.) Dacă tabelul nu poate fi reparat, atunci restaurați-l din copiile dumneavoastră de siguranță. Instrucțiunile pentru refacerea tabelelor folosind fișierele copie de siguranță și jurnalele de actualizare se găsesc în capitolul 11.

Efectuarea reparației standard a unui tabel

Pentru a repara un tabel, parcurgeți următoarele etape:

1. Încercați să remediați tabelul folosind opțiunea `--recover`, dar folosiți și opțiunea `--quick` pentru a încerca să refaceți tabelul numai în funcție de conținutul fișierului index. Astfel, tabelul va fi reparat fără nici o interacțiune cu fișierul de date:

```
% myisamchk --recover --quick nume_tabel
% isamchk --recover --quick nume_tabel
```

2. Dacă problemele persistă, încercați comanda prezentată în etapa anterioară, dar omiteți opțiunea `--quick`, pentru a permite utilitarelor `myisamchk` și `isamchk` să modifice și fișierul de date:

```
% myisamchk --recover nume_tabel
% isamchk --recover nume_tabel
```

3. Dacă această metodă nu reușește, încercați modul de reparare `--safe-recover`. Acesta este mai lent decât modul de refacere obișnuit, dar este capabil de a remedia unele probleme pe care modul `--recover` nu le poate rezolva:

```
% myisamchk --safe-recover nume_tabel
% isamchk --safe-recover nume_tabel
```

Dacă utilitarele `myisamchk` sau `isamchk` se opresc în timpul oricăreia din aceste etape, cu un mesaj de eroare de forma `"Can't create new temp file: nume_fisier"` (nu poate crea un nou fișier temporar: *nume_fișier*), trebuie să repetați comanda și să adăugați opțiunea `--force`, pentru a forța eliminarea fișierului temporar. Probabil că acesta a fost „uitat" de la o încercare de reparare anterioară.

Copiați tabelele înainte de a le repara

O măsură generală de precauție pe care trebuie să o luați înainte de a efectua repararea tabelelor este de a efectua copii ale fișierelor tabel, în eventualitatea în care lucrurile scapă de sub control. Este puțin probabil să se întâmple așa ceva, dar *dacă* se întâmplă, puteți crea o nouă copie a tabelului din fișierele copiate și puteți încerca o altă metodă de refacere.

Ce este de făcut când metoda standard de reparare a tabelelor nu reușește

Dacă procedura de reparare standard nu reușește să repare tabelul, este posibil ca fișierul index să lipsească sau să fie imposibil de reparat. De asemenea, este posibil, deși puțin probabil, ca fișierul de descriere a tabelului să fie inexistent. În ambele cazuri va trebui să înlocuiți fișierele afectate, după care încercați din nou procedura de reparare standard.

Pentru a regenera fișierul index, folosiți următoarea procedură:

1. Deplasați-vă la catalogul de baze de date care conține tabelul deteriorat.
2. Deplasați fișierul de date al tabelului într-un loc sigur.
3. Invocați `mysql` și creați un nou tabel vid prin executarea următoarei instrucțiuni, care folosește fișierul de descriere a tabelului `nume_tabel.frm` pentru a regenera noul fișier de date, respectiv noul fișier index, pornind de la zero:

   `mysql DELETE FROM nume_tabel;`
4. Părăsiți programul `mysql` și deplasați fișierul de date original înapoi în catalogul de baze de date, înlocuind noul fișier de date vid pe care l-ați creat.
5. Încercați din nou să reparați tabelul prin metoda standard.

Pentru a reface fișierul de descriere a tabelului, restaurați-l pornind de la fișierele copie de siguranță, apoi încercați din nou procedura de reparare standard. Dacă, dintr-un motiv sau altul, nu dispuneți de o copie de siguranță, dar cunoașteți instrucțiunea `CREATE TABLE` care trebuie folosită pentru a crea tabelul, atunci mai aveți o șansă de a repara tabelul:

1. Deplasați-vă la catalogul de baze de date care conține tabelul deteriorat.
2. Deplasați fișierul de date al tabelului într-un loc sigur. Deplasați și fișierul index, dacă doriți să încercați să-l folosiți.
3. Invocați programul `mysql` și emiteți instrucțiunea `CREATE TABLE` care creează tabelul.
4. Părăsiți programul `mysql` și deplasați fișierul de date original înapoi în catalogul de baze de date, înlocuind noul fișier de date pe care l-ați creat. Dacă ați deplasat fișierul index în etapa 2, mutați fișierul index înapoi în catalogul de baze de date.
5. Încercați din nou să reparați tabelul prin metoda standard.

Evitarea interacțiunilor cu serverul MySQL

Când rulați un utilitar de verificare-reparare a tabelelor, nu este de dorit ca serverul MySQL și utilitarul să obțină simultan acces la un tabel. Evident că nu este bine deloc ca ambele programe să scrie în tabel, dar nu este mai bine nici dacă un program citește, iar celălalt scrie. Programul care citește poate fi derutat dacă tabelul este modificat în același timp de către celălalt program.

Puteți garanta inexistența oricărei interacțiuni între server și `myisamchk` sau `isamchk` dacă opriți serverul. Administratorii sunt însă reticenți (ceea ce este de înțeles) în ceea ce privește scoaterea completă din funcțiune a serverului, deoarece astfel devin indisponibile și bazele de date și tabelele fără defecțiuni. Procedurile descrise în această secțiune vă vor ajuta să evitați interacțiunea dintre server și `myisamchk` și `isamchk`.

Un server dispune de două categorii de blocări. Blocarea internă se folosește pentru a împiedica interferența dintre cererile clienților, de exemplu pentru a împiedica întreruperea interogării `SELECT` a unui client de către interogarea `UPDATE` a unui alt client. De asemenea, serverul folosește blocarea externă (blocare la nivelul sistemului de fișiere) pentru a preveni modificarea fișierelor tabel de către alte programe în timp ce fișierele respective sunt folosite de către server. În mod normal, serverul folosește blocarea externă pentru a coopera cu `myisamchk` și `isamchk` în timpul operațiilor de verificare a

tabelelor. Totuși, blocarea externă este dezactivată pe unele sisteme, deoarece nu funcționează în mod fiabil. Procedura pe care trebuie să o alegeți pentru rularea programelor `myisamchk` și `isamchk` depinde de utilizarea sau nu a blocării externe de către server. Dacă serverul nu folosește blocarea externă, trebuie să folosiți un protocol de blocare internă.

Blocarea externă este dezactivată dacă serverul rulează cu opțiunea `--skip-locking`. Această opțiune este prestabilită pe unele sisteme, precum Linux. Puteți determina dacă serverul este sau nu capabil să folosească blocarea externă prin rularea comenzii `mysqladmin variables`. Verificați valoarea variabilei `skip-locking` și procedați după cum urmează:

- Dacă variabila `skip-locking` este dezactivată, blocarea externă este activată. Puteți folosi oricare din cele două utilitare pentru verificarea tabelelor. Serverul și utilitarul vor coopera în vederea accesului la tabele. Totuși, trebuie să evacuați zona cache pentru tabele cu ajutorul comenzii `mysqladmin flush-tables` înainte de a rula vreunul din utilitare. Pentru repararea tabelelor, trebuie să folosiți protocolul de blocare pentru repararea tabelelor.
- Dacă variabila `skip-locking` este activată, blocarea externă este dezactivată și serverul nu va ști dacă `myisamchk` sau `isamchk` verifică sau repară un tabel. Cel mai indicat este să opriți serverul. Dacă insistați să lăsați serverul în funcțiune, trebuie să vă asigurați că nici un client nu obține acces la tabel în timp ce dumneavoastră folosiți tabelul. Nu uitați să folosiți un protocol de blocare adecvat, pentru a indica serverului să lase tabelul în pace și să blocheze accesul clienților la tabel.

Protocoalele de blocare descrise aici folosesc mecanismul intern de blocare al serverului, pentru a-l împiedica pe acesta să obțină accesul la un tabel în timp ce dumneavoastră lucrați la tabelul respectiv cu `myisamchk` sau `isamchk`. Ideea generală este că invocați `mysql` și emiteți o instrucțiune `LOCK TABLE` pentru tabelul pe care doriți să-l verificați sau să-l reparați. Apoi, cu programul `mysql` în stare de inactivitate (adică rulează, dar nu face decât să mențină tabelul blocat), rulați `myisamchk` sau `isamchk`. După ce aceste utilitare și-au încheiat execuția, puteți reveni la sesiunea programului `mysql` și anulați blocarea, pentru a indica serverului că poate folosi din nou tabelul.

Protocoalele de blocare pentru verificare și reparare sunt oarecum diferite. Pentru verificare, trebuie să obțineți numai o blocare pentru citire. În acest caz, citiți tabelul, dar nu-l modificați, deci puteți permite și altor clienți să-l citească. Blocarea pentru citire este suficientă pentru a-i împiedica pe alți clienți să modifice tabelul. Pentru reparare, trebuie să vă procurați o blocare pentru scriere, cu scopul de a-i împiedica pe alți clienți să modifice tabelul în timp ce dumneavoastră lucrați la tabel.

Protocoalele de blocare folosesc instrucțiunile `LOCK TABLE` și `UNLOCK TABLE` pentru a dobândi, respectiv pentru anula blocările. De asemenea, protocoalele folosesc `FLUSH TABLES` pentru a evacua pe disc toate modificările suspendate și pentru a redeschide tabelul după ce acesta a fost modificat de un utilitar de reparare a tabelului. *Trebuie* să efectuați toate instrucțiunile `LOCK`, `FLUSH` și `UNLOCK` în cadrul unei singure sesiuni `mysql`. Dacă blocați un tabel și apoi părăsiți programul `mysql`, blocarea va fi anulată și rularea utilitarelor `myisamchk` și `isamchk` nu se va desfășura în condiții de siguranță!

Procedurile de blocare se pot desfășura cel mai ușor dacă păstrați deschise două ferestre: una pentru rularea programului `mysql` și alta pentru rularea programelor `myisamchk` sau `isamchk`, ceea ce vă permite să comutați cu ușurință de la un program la altul. Dacă nu lucrați într-un mediu cu ferestre, trebuie să suspendați și să reluați execuția programului `mysql` folosind funcționalitățile de control al sarcinilor caracteristice interpretorului în timp ce rulați `myisamchk` sau `isamchk`. Instrucțiunile următoare prezintă comenzile pentru `myisamchk` și `isamchk`; folosiți-le pe cele care sunt adecvate pentru tabelul sau tabelele la care lucrați.

Blocarea unui tabel în vederea operațiilor de verificare

Această procedură se folosește numai pentru verificarea unui tabel, nu pentru repararea acestuia. În fereastra 1, invocați `mysql` și emiteți următoarele instrucțiuni:

```
% mysql nume_baza_de_date
mysql> LOCK TABLE nume_tabel READ;
mysql> FLUSH TABLES;
```

Blocarea împiedică pe alți clienți să scrie în tabel și să modifice tabelul în timp ce dumneavoastră îl verificați. Instrucțiunea `FLUSH` determină serverul să închidă fișierele tabel, ceea ce determină evacuarea tuturor modificărilor care mai pot exista în cache.

Cu programul `mysql` inactiv, comutați în fereastra 2 și verificați tabelul:

```
% myisamchk nume_tabel
% isamchk nume_tabel
```

Când `myisamchk` sau `isamchk` și-a încheiat execuția, reveniți la sesiunea `mysql` în fereastra 1 și anulați blocarea tabelului:

```
mysql> UNLOCK TABLE;
```

Dacă `myisamchk` sau `isamchk` raportează identificarea unei probleme cu tabelul, va trebui să efectuați repararea tabelului.

Blocarea unui tabel în vederea operațiilor de reparare

Procedura de blocare pentru repararea unui tabel este similară cu procedura pentru verificarea acestuia, cu două diferențe. Mai întâi, trebuie să obțineți o blocare pentru scriere, nu una pentru citire. Veți modifica tabelul, deci nu puteți permite sub nici o formă clienților accesul la acesta. În al doilea rând, trebuie să emiteți o instrucțiune `FLUSH TABLES` după efectuarea reparațiilor, deoarece `myisamchk` și `isamchk` construiesc un nou fișier index, dar serverul nu va remarca acest lucru decât dacă evacuați din nou zona cache pentru tabele. Pentru a executa această procedură, invocați `mysql` în fereastra 1 și emiteți următoarele instrucțiuni:

```
% mysql nume_baza_de_date
mysql> LOCK TABLE nume_tabel WRITE;
mysql> FLUSH TABLES;
```

Cu programul `mysql` inactiv, faceți copii ale fișierelor tabel, dacă nu ați procedat deja astfel, apoi rulați `myisamchk` sau `isamchk`:

```
% cp nume_tabel.* /un/alt/catalog
% myisamchk --recover nume_tabel
% isamchk --recover nume_tabel
```


Opțiunea `--recover` nu are decât un caracter ilustrativ. Opțiunile particulare pe care le alegeți vor depinde de tipul de reparație pe care îl executați. După ce programele `myisamchk` sau `isamchk` și-au încheiat execuția, reveniți la sesiunea `mysql` din fereastra 1, evacuați din nou zona de memorie cache pentru tabele și anulați blocarea tabelului:

```
mysql> FLUSH TABLES;
mysql> UNLOCK TABLE;
```

Mărirea vitezei de lucru a programelor `myisamchk` și `isamchk`

Rularea programelor `myisamchk` și `isamchk` poate necesita un timp îndelungat, mai ales dacă lucrați cu un tabel mare sau folosiți una dintre metodele mai amănunțite de verificare sau de reparare. Puteți mări viteza de lucru a acestor programe, indicându-le să folosească o cantitate mai mare de memorie atunci când rulează.

Reducerea timpului de inactivitate a serverului

O altă modalitate de a împiedica serverul să obțină accesul la tabelul unde lucrați este de a utiliza copii ale fișierelor tabel, amplasate în afara catalogului de date. Totuși, problema cu interacțiunea nu este eliminată, deoarece trebuie în continuare să împiedicați serverul să obțină accesul la tabel (și eventual chiar să-l modifice!) în timp ce dumneavoastră efectuați copiile. Totuși, aceasta este o metodă de a reduce timpul de inactivitate a serverului și poate fi interesantă pentru dumneavoastră, dacă sunteți reticent în a scoate complet serverul din funcțiune. Opriți serverul pentru o scurtă perioadă de timp, atunci când copiați fișierele tabel de care aveți nevoie într-un alt catalog, după care reporniți serverul.

Planuri de viitor pentru `myisamchk`

Se intenționează ca funcțiile de verificare și reparare a tabelelor de care dispune utilitarul `myisamchk` să fie fuzionate în server într-un anumit moment din dezvoltarea seriei de versiuni MySQL 3.23. Când acest lucru se va întâmpla, verificarea și repararea tabelelor vor deveni mult mai simple, deoarece nu vor mai apărea problemele legate de interacțiunea serverului cu programul `myisamchk`.

De asemenea, veți putea cere serverului să verifice tabelele la pornire, deci nu va fi necesară stabilirea unor comenzi speciale ce ar urma să fie executate la încărcarea sistemului de operare, înainte de pornirea serverului. Totuși, această situație nu este valabilă pentru tabelele ISAM, deci, atunci când serverul va fi dotat cu funcționalitățile de verificare și reparare, v-aș recomanda să aveți în vedere conversia tabelelor ISAM în tabele MyISAM. Examinați manualul de referință MySQL al noilor versiuni, pentru a vedea evoluțiile care au loc în acest domeniu. Puteți converti un tabel folosind o instrucțiune `ALTER TABLE`:

```
ALTER TABLE nume_tabel TYPE = MYISAM
```

Ambele utilitare dispun de numeroși parametri de operare care pot fi configurați. Cele mai importante din aceste variabile controlează dimensiunile bufferelor folosite de către programe:

Variabilă	Semnificație
`key_buffer_size`	Dimensiunea bufferului folosit pentru stocarea blocurilor de index
`read_buffer_size`	Dimensiunea bufferului folosit pentru operațiile de citire
`sort_buffer_size`	Dimensiunea bufferului folosit pentru sortare
`write_buffer_size`	Dimensiunea bufferului folosit pentru operațiile de scriere

Pentru a afla valorile folosite în mod prestabilit de fiecare program pentru aceste variabile, rulați programul cu opțiunea --help. Pentru a specifica o altă valoare, folosiți --set-variable *variabila=valoare* sau -O *variabila=valoare* în linia de comandă. Puteți prescurta numele variabilelor sub forma key, read, sort și write. De exemplu, pentru a indica programului myisamchk să folosească un buffer de sortare de 16MB și buffere de citire, respectiv scriere, de 1MB, invocați programul astfel:

```
% myisamchk -O sort=16M -O read=1M -O write=1M ...
```

Variabila sort_buffer_size este folosită numai cu opțiunea --recover (nu cu opțiunea --safe-recover), caz în care variabila key_buffer nu este folosită.

Planificarea întreținerii preventive

Trebuie să aveți în vedere alcătuirea unei planificări a operațiilor de întreținere preventivă, pentru a contribui la detectarea automată a problemelor, astfel încât să puteți lua măsuri de remediere a acestora:

1. Executați sistematic copii de siguranță ale bazelor de date și activați caracteristica de consemnare în jurnalele de actualizare. Nu-i așa c-am mai spus-o?
2. Stabiliți sistematic o verificare periodică a tabelelor. Prin verificarea tabelelor dumneavoastră, veți diminua posibilitatea de a fi obligat să vă folosiți copiile de siguranță. Această operație se poate executa cel mai simplu sub forma unei sarcini a utilitarului cron, invocată în mod caracteristic din fișierul crontab al contului folosit pentru rularea serverului. De exemplu, dacă rulați serverul ca utilizator mysqladm, atunci puteți stabili o verificare periodică din fișierul crontab pentru mysqladm. Dacă nu știți să folosiți cron, examinați paginile aferente din manualul UNIX folosind aceste comenzi:

```
% man cron
% man crontab
```

3. Verificați tabelele din bazele de date în timpul procedurii de încărcare a sistemului de operare, înainte de pornirea serverului. S-ar putea ca reîncărcarea sistemului de operare să aibă loc datorită unei căderi anterioare a sistemului. În acest caz, este posibil ca tabelele din baza de date să fie deteriorate și trebuie verificate.

Pentru a executa o verificare automată a tabelelor, puteți scrie un script care comută în catalogul de date al serverului și rulează myisamchk și isamchk pe toate tabelele din baza dumneavoastră de date. În scripturile prezentate în expunerea următoare, vom folosi ambele programe. Dacă dispuneți numai de tabele MyISAM sau numai de tabele ISAM, aveți nevoie numai de unul din aceste programe și îl puteți omite pe celălalt din scripturi.

Atât myisamchk, cât și isamchk generează date de ieșire în modul de verificare a tabelelor, pentru a vă informa care sunt tabelele pe care le verifică, chiar dacă nu există nici un fel de probleme. Pentru intrările crontab, în mod normal este de dorit ca datele de ieșire să fie suprimate, cu excepția situațiilor când apar erori în tabele. (În mod caracteristic, sarcinile utilitarului cron generează un mesaj de poștă dacă vreo sarcină produce date de ieșire, și nu există prea multe motive să primiți mesaje pentru sarcini de verificare a tabelelor care nu găsesc nici o problemă.) Dacă invocați orice utilitar cu opțiunea --silent, va genera date de ieșire numai la apariția problemelor. De asemenea, myisamchk acceptă o opțiune --fast, care îi permite să omită orice tabel care nu a fost modificat de la ultima verificare.

Un script simplu, care verifică toate tabelele din catalogul de date al serverului, se prezintă astfel (DATADIR trebuie înlocuit cu o valoare adecvată pentru sistemul dumneavoastră):

```
#! /bin/sh
cd DATADIR
myisamchk --silent --fast t */*.MYI
isamchk --silent */*.ISM
```

O potențială problemă a acestui script este aceea că, dacă dispuneți de numeroase tabele modelele cu caractere de înlocuire */*.MYI sau */*.ISM pot agasa interpretorul cu eror de genul „prea multe argumente". Un script alternativ poate fi scris astfel (din nou înlocuiți DATADIR cu o valoare adecvată pentru sistemul dumneavoastră):

```
#! /bin/sh
datadir=DATADIR
find $datadir -name "*.MYI" -print | xargs myisamchk --silent --fast
find $datadir -name "*.ISM" -print | xargs isamchk --silent
```

Indiferent de forma de script pentru care optați, voi presupune că îl veți denum check_mysql_tables. Nu uitați să-i modificați modul, astfel încât scriptul să fie executabi Apoi, trebuie să invocați scriptul manual, pentru a verifica dacă funcționează corect:

```
% chmod +x check_mysql_tables
% check_mysql_tables
```

În mod ideal, nu vor exista date de ieșire. Dacă sistemul dumneavoastră nu acceptă blocare externă, este posibil ca serverul să modifice un tabel în timp ce îl verificați. În acest caz, scrip tul poate raporta probleme pentru tabele unde aceste probleme nu există. Situația nu este pre fericită, dar este de preferat cazului invers, când scriptul nu raportează probleme, dar aceste există. Dacă sistemul dumneavoastră acceptă blocarea externă, această problemă nu va apăre

Secțiunile următoare prezintă modul de configurare a scriptului astfel încât acesta să f executat automat de cron și la pornirea sistemului. În exemplele din aceste secțiuni, v presupune că scriptul este instalat în catalogul /usr/users/mysqladm/bin. Va trebui modificați calea conform configurației sistemului dumneavoastră.

Dacă rulați mai multe servere în sistemul dumneavoastră, va trebui să modificați proc dura astfel încât să verificați tabelele din catalogul de date al fiecărui server. Puteți fa aceasta executând copii diferite ale scriptului check_mysql_tables sau prin modificar scriptului astfel încât să accepte un argument de linie de comandă care să specifice cat logul de date pe care doriți să-l verificați.

Verificarea periodică a tabelelor cu ajutorul utilitarului cron

Să presupunem că doriți să invocați scriptul check_mysql_tables din fișierul cront pentru utilizatorul mysqladm. Deschideți sesiunea de lucru sub contul respectiv, apoi ed tați fișierul dumneavoastră crontab folosind această comandă:

```
% crontab -e
```

Această comandă vă va duce într-un editor de texte, în care va insera o copie a fișierul crontab curent (care poate fi vid, dacă nu ați mai executat această operație în prealabi Adăugați la fișier o linie care se prezintă astfel:

```
0 3 * * 0 /usr/users/mysqladm/bin/check_mysql_tables
```

Această intrare indică utilitarului cron să ruleze comanda în fiecare duminică, la ora 3 dimineața. Puteți modifica ora sau planificarea după dorință. Căutați în pagina de manual crontab pentru a determina formatul intrărilor.

Verificarea tabelelor la pornirea sistemului

Dacă folosiți un sistem de tip BSD și ați adăugat comanda de pornire a serverului la fișierul /etc/rc.local sau la un echivalent al acestuia, puteți pur și simplu invoca check_mysql_tables din fișierul respectiv, înainte de a porni serverul.

Dacă folosiți metoda de pornire caracteristică sistemelor de tip System V, care invocă mysql.server din unul din cataloagele /etc/rc.d, procedura este ceva mai complexă. Scripturile de pornire din aceste cataloage trebuie să înțeleagă argumentele start și stop, astfel încât să poată întreprinde acțiuni adecvate la pornirea, respectiv oprirea sistemului. Pentru a executa verificarea tabelelor, putem scrie un script care invocă check_mysql_tables atunci când argumentul este start și care nu face nimic atunci când argumentul este stop. Să denumim scriptul mysql.check; aspectul său este următorul:

```
#! /bin/sh

# Sa vedem cum am fost apelati.
case "$1" in
  start)
    echo -n "Verifica tabelele MySQL: "
    if [ -x /usr/users/mysqladm/bin/check_mysql_tables ]; then
         /usr/users/mysqladm/bin/check_mysql_tables
    fi
    ;;
  stop)
    # nu face nimic
    ;;
  *)
    echo "Usage: $0 {start|stop}"
    exit 1
esac

exit 0
```

Acum, puteți instala mysql.check folosind o procedură similară celei descrise în capitolul 11 pentru instalarea scriptului mysql.server. Nu uitați să atribuiți scriptului mysql.check un număr de prefix inferior în catalogul la nivel de rulare, astfel încât să ruleze înainte de mysql.server. De exemplu, dacă vă legați la mysql.server ca S99mysql.server din catalogul la nivel de rulare, vă puteți lega la mysql.check ca S98mysql.check.

PARTEA A IV-A

Anexe

A Obținerea și instalarea programelor
B Referință de tipuri de coloane
C Referință de operatori și funcții
D Referință de sintaxă SQL
E Referință de programe MySQL
F Referință API C
G Referință API Perl DBI
H Referință API PHP
I Instrumente utile produse de terțe părți
J Furnizori de servicii Internet

ANEXA A

Obținerea și instalarea programelor

Această anexă descrie modul de obținere și instalare a programelor de care veți avea nevoie pentru a rula MySQL și programele corelate create de terțe părți descrise în această carte, precum suportul pentru Perl DBI și CGI.pm, PHP și Apache. De asemenea, este prezentat modul de obținere a fișierelor asociate cu baza de date demonstrativă `samp_db` folosită pe parcursul acestei cărți.

Scopul acestui capitol este de a aduna la un loc instrucțiunile de instalare pentru toate pachetele, nu de a înlocui instrucțiunile incluse în fiecare pachet. De fapt, vă recomand să citiți aceste instrucțiuni. Anexa de față furnizează instrucțiuni de ordin general care ar trebui să fie suficiente în majoritatea situațiilor, dar fiecare distribuție software conține și instrucțiuni care vă ajută să remediați problemele în cazul eșecului unei proceduri de instalare standard. De exemplu, manualul MySQL conține un capitol de mari dimensiuni dedicat procedurilor de instalare și care include soluții pentru numeroase probleme specifice sistemului.

Selectarea programelor de instalat

Trebuie să instalați MySQL dacă nu ați procedat deja astfel, dar este necesar să instalați numai acele instrumente create de terțe părți pe care doriți să le utilizați:

- Dacă doriți să scrieți scripturi Perl care obțin acces la bazele de date MySQL, trebuie să instalați suportul MySQL pentru programarea clienților și modulele DBI. Dacă intenționați să scrieți scripturi DBI care furnizează acces prin Web la bazele dumneavoastră de date, probabil că veți dori să instalați modulul CGI.pm și veți avea nevoie de un server Web. În această carte este folosit serverul Apache, dar se pot folosi și alte servere.
- Dacă doriți să scrieți scripturi PHP, trebuie să instalați suport de programare a clienților și PHP. În mod normal, PHP este folosit pentru activitatea de scripting în Web, ceea ce înseamnă că mai aveți nevoie și de un server Web. Serverul Apache este folosit în această carte deoarece Apache are suport modular pentru PHP. Aceasta înseamnă că puteți instala PHP ca parte încorporată a binarului dumneavoastră Apache. Dacă nu puteți modifica exemplarul dumneavoastră al serverului Apache (de exemplu dacă vă aflați pe o gazdă ISP și ISP controlează Apache) sau dacă rulați un alt server Web, puteți construi PHP ca program autonom și rulați scripturi PHP sub formă de scripturi CGI standard. Instalarea PHP ca modul Apache asigură performanțe mai bune decât rularea acestuia ca program autonom.

Fișierele binare precompilate sunt disponibile pentru multe dintre pachetele de instalare. Pentru Linux sunt disponibile mai multe fișiere RPM. Dacă preferați să compilați programe de la sursă sau dacă nu este disponibilă o distribuție binară pentru platforma dumneavoastră, veți avea nevoie de un compilator C (C++ pentru MySQL). Dezvoltatorii MySQL recomandă utilizarea compilatorului `gcc` 2.95.1 sau a unei versiuni mai recente a acestuia pentru a construi MySQL. De asemenea, se pot folosi și versiunile compilatorului `egcs` începând de la numărul 1.03.

Alte instrumente de care veți avea nevoie includ `gunzip` și `tar` (dacă instalați din fișiere `tar` comprimate), respectiv `rpm` (dacă instalați din fișiere RPM).

Obținerea programelor

Principalele puncte de distribuție pentru fiecare dintre pachetele de care veți avea nevoie sunt prezentate în tabelul următor. Unele pachete pot fi obținute din mai multe locații:

Pachet	Locație
MySQL	`http://www.mysql.com/` `ftp://ftp.mysql.com/pub/mysql/`
DBI	`http://www.symbolstone.org/technology/perl/DBI/`
	`http://www.mysql.com/Contrib/`
PHP	`http://www.php.net/`
	`ftp://ftp.php.net/`
Apache	`http://www.apache.org/` `ftp://ftp.apache.org/`
CGI.pm	`http:/stein.cshl.org/WWW/software/CGI/`

Dacă folosiți FTP pentru transferul fișierelor, nu uitați să le transferați în mod binar, pentru a evita deteriorarea fișierelor.

Există multe situri oglindă pentru MySQL. `www.mysql.com` se află în Scandinavia, deci, dacă alegeți un sit oglindă care este mai apropiat de dumneavoastră, probabil că veți obține timpi de descărcare mai buni. Siturile oglindă sunt menționate la următoarea adresă:

`http://www.mysql.com/mirrors.html`

Pentru programele Perl, puteți folosi întotdeauna `http://cpan.perl.org/` pentru a localiza siturile oglindă care conțin modulele de care aveți nevoie.

Selectarea versiunii care va fi instalată

Versiunea unui pachet pe care îl instalați depinde de necesitățile dumneavoastră:

- Dacă aveți nevoie de un maximum de stabilitate, probabil că trebuie să fiți conservator și să folosiți cea mai recentă versiune stabilă (nu beta) a unui pachet. Astfel, beneficiați de cele mai recente caracteristici și de cel mai mare număr de remedii ale hibelor, fără a vă expune programelor experimentale din versiunea beta.
- Dacă doriți să fiți mereu la modă sau dacă sunteți programator, poate doriți să folosiți cea mai recentă versiune beta. Desigur, dacă aveți nevoie în mod absolut de o caracteristică existentă numai într-o versiune beta, nu aveți altă opțiune decât să folosiți versiunea respectivă.

- Pentru MySQL, distribuțiile RPM și binare încorporate sunt deseori construite folosind indicatori de optimizare, care sunt mai buni decât ceea ce poate determina, de unul singur, scriptul de configurare din distribuția sursă. De asemenea, dezvoltatorii MySQL construiesc distribuții folosind unele compilatoare comerciale de optimizare, pentru a mări și mai mult viteza sistemului MySQL. În consecință, programele din aceste categorii de distribuții pot rula mai rapid decât cele pe care le compilați dumneavoastră. Dacă doriți să obțineți un maximum de performanță, puteți dori să încercați o distribuție încorporată.

Siturile Web pentru fiecare pachet indică versiunile care constituie cele mai recente variante stabile, precum și cele care reprezintă variante beta. De asemenea, aceste situri conțin liste cu modificările caracteristicilor pentru fiecare versiune în parte, pentru a vă ajuta să decideți care este varianta cea mai potrivită pentru dumneavoastră.

Dacă lucrați cu distribuții RPM sau binare, despachetarea unei distribuții este echivalentă cu instalarea acesteia, deoarece fișierele sunt despachetate în cataloagele unde doriți să se găsească acestea. Dacă o distribuție instalează fișiere în cataloage protejate, atunci trebuie să lucrați sub contul `root` pentru a despacheta distribuția.

Pentru distribuțiile sursă, le puteți despacheta într-o regiune pe care o folosiți pentru compilare și apoi instalați programul în locațiile de instalare. Pentru a efectua etapa de instalare este necesar să lucrați sub contul `root`, dar această calitate nu este necesară pentru nici una dintre etapele de configurare sau compilare.

Înscrierea în listele de corespondență pentru a obține ajutor

Când instalați un pachet, se recomandă să vă înscrieți pe lista de discuții generală pentru pachetul respectiv, astfel încât să puteți pune întrebări și primi răspunsuri utile. Dacă instalați variante beta, categoric trebuie să vă pregătiți pentru a vă înscrie în lista de corespondență asociată programului și pentru a citi mesajele din această listă, pentru a fi la curent cu rapoartele privind hibele și cu remediile apărute. Dacă nu vă înscrieți într-o listă de discuții generală, trebuie să vă înscrieți cel puțin în lista de anunțuri, pentru a primi înștiințări privind noile variante. În introducerea acestei cărți se găsesc instrucțiuni pentru înscrierea pe listele de corespondență și pentru utilizarea acestora. Siturile Web pentru fiecare pachet furnizează de asemenea informații privind înscrierea.

Dacă instalați de la o distribuție sursă, numeroase pachete sunt configurate cu ajutorul utilitarului `configure`, ceea ce facilitează configurarea și construirea programelor pe o varietate de sisteme. În cazul în care construirea unui program eșuează, puteți fi obligat să rulați din nou utilitarul `configure` cu alte opțiuni decât cele pe care le-ați specificat inițial. Dacă este necesar, trebuie să împiedicați utilitarul `configure` să folosească informații salvate de la rularea anterioară a acestui program. Puteți șterge astfel informațiile stocate:

```
% make distclean
```

Alternativ, mai puteți proceda și astfel:

```
% rm config.cache
% make clean
```

Indiferent de caz, trebuie să rulați din nou `configure` după aceea.

Instalarea sistemului MySQL pe UNIX

Există distribuții disponibile pentru diferite versiuni de MySQL. Variantele stabile curente au numere de versiune din seria 3.22. Variantele de dezvoltare curente se află în seria 3.23. În general, trebuie să folosiți versiunea cu numărul cel mai mare din seria pe care doriți să o folosiți disponibilă în formatul de distribuție pe care doriți să-l utilizați.

Distribuțiile MySQL se folosesc în format binar, RPM și format sursă. Distribuțiile binară și RPM sunt mai ușor de instalat, dar trebuie să acceptați parametrii prestabiliți pentru structura și configurarea instalării, care sunt încorporați în distribuție. Distribuțiile sursă sunt mai dificil de instalat, deoarece trebuie să compilați programul, dar dobândiți un control mai mare asupra parametrilor de configurare. De exemplu, puteți compila distribuția destinată numai suportului pentru clienți fără a construi serverul, iar dumneavoastră puteți modifica locația unde doriți să instalați programul.

Distribuțiile conțin una sau mai multe dintre următoarele componente:

- Serverul `mysqld`
- Programele client (`mysql`, `mysqladmin` și așa mai departe) și suport pentru programarea clienților (fișiere bibliotecă și antet)
- Documentație
- Baza de date pentru evaluarea performanțelor
- Suport pentru limbaj

Distribuțiile sursă și binară conțin toate aceste elemente. Fiecare fișier RPM conține numai câteva dintre ele, deci puteți fi obligat să instalați mai multe fișiere RPM pentru a obține tot ceea ce aveți nevoie.

Dacă intenționați să vă conectați la un server care rulează deja pe un alt calculator, nu trebuie să instalați un server. Dar trebuie întotdeauna să instalați programe client:

- Dacă nu rulați un server, veți avea nevoie de clienți pentru a vă putea conecta la un server situat pe un alt calculator.
- Dacă rulați un server, doriți să vă puteți conecta la acesta de la gazda serverului, nu să fiți forțat să deschideți sesiunea de lucru la un alt calculator care dispune de programul client, numai pentru a vă testa serverul.

O trecere în revistă a instalării sistemului MySQL

Instalarea sistemului MySQL pe UNIX implică următoarele etape:

1. Creați un cont UNIX pentru utilizatorul și grupul sub al cărui cont va rula utilizatorul (dacă instalați serverul).
2. Obțineți și despachetați toate distribuțiile pe care doriți să le instalați. Dacă folosiți o distribuție sursă, compilați-o și instalați-o.
3. Rulați scriptul `mysql_install_db` pentru a inițializa catalogul de date și tabelele de acordare (numai când sistemul MySQL este instalat pentru prima dată pe un calculator).
4. Porniți serverul.

5. Citiți capitolul 11, „Administrarea generală a sistemului MySQL", pentru a vă familiariza cu procedurile administrative generale. În particular, trebuie să citiți secțiunile cu privire la pornirea și oprirea serverului, precum și la rularea serverului în calitate de utilizator fără privilegii.

Crearea unui cont pentru utilizatorul MySQL

Puteți omite această secțiune dacă rulați numai programe client MySQL și nu intenționați să rulați un server MySQL.

Serverul MySQL poate fi rulat sub orice cont de utilizator UNIX pe sistemul dumneavoastră, dar, din motive administrative și de securitate, cel mai bine este să nu rulați serverul sub contul `root`. Vă recomand să creați un cont separat pentru activitatea de administrare a sistemului MySQL și să rulați serverul sub contul respectiv. În acest mod, puteți deschide sesiunea de lucru sub contul utilizatorului respectiv și aveți privilegii complete în catalogul de date pentru executarea operațiilor de întreținere și depanare. Procedurile pentru crearea conturilor de utilizator variază de la un sistem la altul. Consultați-vă documentația locală, pentru a obține detalii mai concrete.

Această carte folosește `mysqladm` și `mysqlgrp` ca nume de utilizator și nume de grup UNIX ale acestui cont. Dacă intenționați să instalați MySQL numai pentru uz personal, îl puteți rula sub contul propriu, caz în care veți putea folosi propriile dumneavoastră nume de utilizator și de grup oriunde vedeți mențiunile `mysqladm` și `mysqlgrp` în această carte. Dacă instalați din fișiere RPM, un utilizator denumit `mysql` este creat automat ca parte a procedurii de instalare RPM. În acest caz, veți înlocui `mysqladm` cu `mysql`.

Avantajele utilizării unui cont separat, fără privilegii, în locul contului `root` pentru exploatarea sistemului MySQL sunt următoarele:

- Dacă nu rulați MySQL ca `root`, nimeni nu poate exploata serverul ca breșă de securitate pentru a dobândi accesul sub contul `root`.
- Executarea operațiunilor administrative din MySQL din contul unui utilizator fără privilegii este mai sigură decât din contul `root`.
- Serverul va crea fișiere aflate în posesia utilizatorului `mysqladm`, nu în cea a utilizatorului `root`. Cu cât utilizatorul `root` deține mai puține fișiere în sistemul dumneavoastră, cu atât mai bine.
- Din punct de vedere conceptual, este mai indicat ca activitatea legată de MySQL să se desfășoare într-un cont separat; astfel, aspectele legate de MySQL din sistemul dumneavoastră devin mai ușor de sesizat. De exemplu, în catalogul unde sunt păstrate fișierele `cron`, veți avea un fișier separat pentru utilizatorul MySQL `mysqladm`. În caz contrar, sarcinile `cron` aferente sistemului MySQL se vor găsi în fișierul utilizatorului `root`, alături de toate operațiile executate periodic în contul `root`.

Obținerea și instalarea distribuției MySQL

În instrucțiunile următoare, voi folosi `versiune` ca simbol al numărului versiunii distribuției MySQL pe care o utilizați, respectiv `platforma` ca nume al platformei pe care o instalați. Acestea sunt folosite în numele de fișiere din distribuție, astfel încât distribuțiile să poată fi identificate cu ușurință și diferențiate una de alta. Un număr de versiune este ceva de genul `3.22.26` sau `3.23.4-alpha`. Un nume de platformă este ceva de genul `sgi-irix6.3-mips` sau `dec-osf4.0b-alpha`.

Instalarea unei distribuții binare

Fișierele unei distribuții binare au nume precum `mysql-versiune-platforma.tar.gz`. Procurați-vă fișierul de distribuție pentru versiunea și platforma dorite și inserați-l în catalogul unde doriți să instalați MySQL, de exemplu `/usr/local`.

Despachetați distribuția folosind una dintre următoarele comenzi (folosiți a doua comandă, dacă versiunea dumneavoastră a programului `tar` nu înțelege opțiunea `z` pentru decomprimarea distribuției):

```
% tar zxf mysql-versiune-platforma.tar.gz
% gunzip < mysql-versiune-platforma.tar.gz | tar xf -
```

Prin despachetarea distribuției se creează un catalog, denumit `mysql-versiune-platforma`, care include conținutul distribuției. Pentru a facilita referirea la acest catalog, creați o legătură simbolică:

```
% ln -s mysql-versiune-platforma mysql
```

Acum, puteți face referire la catalogul de instalare sub numele `/usr/local/mysql`, dacă ați instalat MySQL în catalogul `/usr/local`.

Dacă veți folosi numai suportul pentru clienți furnizat de distribuție și nu rulați un server, ați terminat de instalat sistemul MySQL. Dacă instalați MySQL pentru prima dată, totuși, consultați secțiunea „Inițializarea catalogului de date și a tabelelor de acordare". Dacă actualizați o instalare existentă, consultați secțiunea „Pornirea serverului".

Instalarea unei distribuții RPM

Fișierele RPM sunt disponibile pentru instalarea sistemului MySQL pe sisteme Linux. Numele de fișier ale acestora sunt următoarele:

- `MySQL-versiune-platforma.rpm`

  Programul server.
- `MySQL-client.versiune-platforma.rpm`

  Programele client.
- `MySQL-devel.versiune-platforma.rpm`

  Suport pentru dezvoltare (biblioteci client și fișiere antet) pentru scrierea programelo client. Veți avea nevoie de acestea dacă doriți să folosiți sau să scrieți scripturi Perl DB pentru accesul la bazele de date MySQL.
- `MySQL-bench.versiune-platforma.rpm`

  Programe de testare și evaluare a performanțelor. Necesită Perl și modulul `Msql-Mysql-modules`. Vă puteți procura un RPM pentru modulul `Msql-Mysql-modules` de la adresa `www.mysql.com`.
- `MySQL-versiune.src.rpm`

  Sursa pentru programul server, programele client și programele de testare și evaluare a performanțelor.

Nu trebuie să vă aflați într-un anumit catalog atunci când instalați MySQL dintr-o distribuție RPM, deoarece fișierele RPM conțin informații care arată unde trebuie instalate fișierele pe care le conțin. Pentru orice fișier RPM `fisier_rpm` puteți afla unde va fi instalat conținutul acestuia cu ajutorul următoarei comenzi:

```
% rpm -qpl fisier_rpm
```

Pentru a instala un fișier RPM, folosiți această comandă:

```
% rpm -i fisier_rpm
```

Diferitele componente ale sistemului MySQL sunt împărțite în diferite fișiere RPM, deci poate fi necesar să instalați mai mult de un fișier RPM. Pentru a instala suportul pentru clienți, folosiți această comandă:

```
% rpm -i MySQL-client-versiune-platforma.rpm
```

Pentru a instala suportul pentru server, folosiți următoarea comandă:

```
% rpm -i MySQL-versiune-platforma.rpm
```

Dacă intenționați să scrieți propriile dumneavoastră programe folosind suportul pentru programarea clienților, nu uitați să instalați fișierul RPM de dezvoltare:

```
% rpm -i MySQL-devel-versiune-platforma.rpm
```

Dacă intenționați să folosiți numai suportul pentru clienți furnizat de distribuție și nu rulați un server, ați terminat de instalat sistemul MySQL. Dacă instalați MySQL pentru prima dată, totuși, consultați secțiunea „Inițializarea catalogului de date și a tabelelor de acordare". Dacă actualizați o instalare existentă, consultați secțiunea „Pornirea serverului".

Dacă doriți să instalați de la fișierul RPM sursă, următoarea comandă este suficientă:

```
% rpm --recompile MySQL-versiune.src.rpm
```

Instalarea unei distribuții sursă

Distribuțiile sursă au nume precum `mysql-versiune.tar.gz`, unde `versiune` este numărul versiunii MySQL. Selectați catalogul unde doriți să despachetați distribuția și deplasați-vă la acesta. Procurați-vă fișierul de distribuție și despachetați-l folosind una dintre următoarele comenzi (folosiți a doua comandă în cazul în care versiunea dumneavoastră a programului `tar` nu înțelege opțiunea z pentru decomprimarea distribuției):

```
% tar zxf mysql-versiune-platforma.tar.gz
% gunzip < mysql-versiune-platforma.tar.gz | tar xf -
```

Prin despachetarea distribuției, se creează un catalog `mysql-versiune`, care include conținutul distribuției. Deplasați-vă în acest catalog:

```
% cd mysql-versiune
```

Acum, trebuie să configurați și să compilați distribuția înainte de a o putea instala. Dacă aceste etape nu dau nici un rezultat, examinați capitolul „Instalarea sistemului MySQL" din manualul de referință MySQL, cu precădere orice note specifice de sistem pe care le poate conține manualul cu privire la tipul calculatorului dumneavoastră.

Folosiți comanda `configure` pentru a configura distribuția:

```
% ./configure
```

Puteți specifica opțiuni pentru `configure`, dacă doriți. Pentru a obține o listă cu opțiunile disponibile, rulați următoarea comandă:


```
% ./configure --help
```

Lista următoare prezintă unele opțiuni de configurare utile pentru mulți:

- `--without-server`

  Configurare numai pentru construirea suportului pentru clienți (programe client sau biblioteci client). Puteți folosi această opțiune dacă intenționați să obțineți acces la un server care rulează deja pe un alt calculator.

- `--prefix=nume_cale`

  În mod prestabilit, catalogul rădăcină folosit la instalare este `/usr/local`. Catalogul de date, clienții, serverul, bibliotecile client și fișierele antet sunt instalate respectiv în cataloagele `var`, `bin`, `libexec`, `lib` și `include` situate în acest catalog. Dacă doriți să modificați catalogul rădăcină de instalare, folosiți opțiunea `--prefix`.

- `--localstatedir=nume_cale`

  Această opțiune modifică amplasamentul catalogului de date. O puteți folosi dacă nu doriți să vă păstrați bazele de date în catalogul `/usr/local/var`.

- `--with-low-memory`

  Fișierul sursă `sql/sql_yacc.cc` necesită o mare cantitate de memorie pentru a fi compilat, ceea ce uneori determină suprasolicitarea construcției. Printre simptomele acestei probleme se numără mesaje de eroare privind „semnalul fatal 11" sau epuizarea memoriei virtuale. Opțiunea `--with-low-memory` determină invocarea compilatorului cu opțiuni care au ca rezultat utilizarea unei cantități mai reduse de memorie.

După ce ați rulat `configure`, compilați distribuția și apoi instalați-o:

```
% make
% make install
```

Dacă nu ați folosit opțiunea `--prefix` pentru a specifica un catalog de instalare în care aveți permisiunea de scriere, este necesar să lucrați sub contul `root` pentru a rula comanda de instalare.

Dacă veți folosi numai suportul pentru clienți furnizat de distribuție și nu rulați un server, ați terminat de instalat sistemul MySQL. Dacă instalați MySQL pentru prima dată, totuși, consultați secțiunea următoare. Dacă actualizați o instalare existentă, consultați secțiunea „Pornirea serverului".

Inițializarea catalogului de date și a tabelelor de acordare

Înainte de a putea folosi sistemul dumneavoastră MySQL instalat, trebuie să inițializați baza de date `mysql`, depozit al tabelelor de acordare care controlează accesul din rețea la serverul dumneavoastră. Această operație este necesară numai pentru instalări noi și numai dacă veți rula un server. Dacă executați numai instalări ale programului client, puteți omite secțiunea de față. Pentru o distribuție binară, rulați comenzile din catalogul de instalare (părintele catalogului `bin` creat de distribuție). Pentru o distribuție sursă, rulați comenzile din catalogul amplasat la nivelul cel mai de sus al distribuției.

`DATADIR` este numele căii de acces spre catalogul dumneavoastră de date. În mod normal, veți rula comenzile următoare din contul `root`. Dacă ați deschis sesiunea de lucru ca

mysqladm sau ați instalat MySQL sub propriul dumneavoastră cont, deoarece intenționați să-l rulați personal, puteți executa comenzile fără a avea calitatea de root și puteți omite comenzile chown și chmod.

Tabelele de acordare prestabilite pot fi configurate rulând scriptul mysql_install_db:[1]

```
# scripts/mysql_install_db
```

Dacă scriptul mysql_install_db eșuează, consultați capitolul „Instalarea sistemului MySQL" din manualul de referință MySQL, pentru a vedea dacă acesta conține informații despre problema pe care o aveți, apoi reluați operația. Rețineți că dacă mysql_install_db nu se derulează complet și cu succes, toate tabelele de acordare pe care le creează sunt potențial incomplete. Trebuie să le eliminați, deoarece mysql_install_db nu va încerca să creeze din nou tabele care descoperă că există deja. Puteți elimina întreaga bază de date mysql astfel:

```
# rm -rf DATADIR/mysql
```

După ce ați rulat scriptul mysql_install_db, închideți serverul dacă instalați o versiune a sistemului MySQL mai veche decât 3.22.10 (în versiunile mai recente, mysql_install_db execută automat această operație):

```
# bin/mysqladmin --user=root shutdown
```

Serverul fiind oprit, modificați utilizatorul proprietar și grupul din care face parte acesta, precum și modul tuturor fișierelor din catalogul de date:

```
# chown -R mysqladm.mysqlgrp DATADIR
# chmod -R go-rwx DATADIR
```

Comanda chown face din utilizatorul MySQL proprietarul fișierelor respective, iar chmod modifică modul pentru a împiedica pe orice utilizator, cu excepția lui mysqladm, să obțină accesul la catalogul de date.

Pornirea serverului

Această etapă este necesară numai dacă veți rula un server. Dacă executați numai instalări ale clienților, puteți omite secțiunea de față. Rulați comenzile din această secțiune din același catalog indicat în secțiunea anterioară. În mod normal, comenzile se rulează din contul root. Dacă ați deschis sesiunea de lucru din contul mysqladm sau ați instalat MySQL sub propriul dumneavoastră cont, puteți rula comenzile fără a fi root și trebuie să omiteți opțiunea --user.

Pentru a porni serverul, folosiți următoarea comandă:

```
# bin/safe_mysqld --user=mysqladm &
```

Opțiunea --user indică serverului să ruleze sub contul mysqladm. Dacă doriți să activați caracteristica de consemnare în jurnal, folosiți în schimb această comandă:

```
# bin/safe_mysqld --user=mysqladm --log &
```

[1] Nu este necesar să executați această operație dacă instalați din fișiere RPM, deoarece mysql_install_db va fi rulat automat pentru dumneavoastră. – N.A.

Instalarea prestabilită permite utilizatorului MySQL root să se conecteze fără parolă. Stabilirea unei parole este o idee bună. De asemenea, trebuie să configurați serverul astfel încât acesta să pornească și să se oprească atunci când sistemul dumneavoastră pornește, respectiv se oprește. De asemenea, este recomandabil să se activeze caracteristica de consemnare în jurnalul de actualizare, deoarece poate fi utilă pentru refacerea datelor. Pentru instrucțiuni cu privire la executarea oricăreia dintre aceste operații, consultați capitolul 11.

Instalarea suportului pentru Perl DBI

Dacă doriți să scrieți scripturi Perl care să obțină acces la baze de date MySQL, instalați programul DBI. Acesta impune instalarea suportului MySQL pentru programarea clienților, deoarece folosește biblioteca client pentru MySQL scrisă în C. De asemenea, dacă doriți să scrieți scripturi DBI bazate pe Web, puteți instala modulul CGI.pm. Programul DBI necesită o versiune rezonabil de recentă a limbajului Perl (5.004 sau ulterioară). Dacă în sistemul dumneavoastră nu este instalat limbajul Perl, vizitați adresa http://www.perl.com/, descărcați o distribuție Perl și instalați-o înainte de a instala suportul pentru DBI.

Suportul pentru DBI necesită trei module, pe care le puteți instala din fișiere sursă sau RPM:

- **Data-Dumper.** Un modul pentru prelucrarea într-un mod convenabil a structurilor de date Perl.
- **DBI.** Principalul driver DBI.
- **Msql-Mysql-modules.** Driverul specific MySQL folosit de DBI atunci când vă conectați la un server MySQL.

Dacă instalați de la sursă, instalați modulele în ordinea prezentată în lista anterioară. În caz contrar, etapa de testare din următoarele instrucțiuni de instalare nu va funcționa.

Instalarea este similară pentru toate cele trei modulele. Când instalați de la sursă, despachetați un fișier de distribuție folosind una dintre următoarele comenzi (folosiți a doua comandă dacă versiunea dumneavoastră a programului tar nu înțelege opțiunea z pentru decomprimarea distribuției):

```
% tar zxf fisier_distributie.tar.gz
% gunzip < fisier_distributie.tar.gz | tar xf -
```

Apoi, deplasați-vă în catalogul de distribuție creat de comanda tar și rulați următoarele comenzi (pentru a rula etapa de instalare, aveți nevoie de contul root):

```
% perl Makefile.PL
% make
% make test
% make install
```

Dacă rulați comenzile precedente pentru distribuția Msql-Mysql-modules, comanda perl vă va pune unele întrebări atunci când generați Makefile:

- **Ce drivere doriți să instalați?** Există opțiuni pentru diferite combinații de MySQL și mSQL. Dacă nu rulați și mSQL, alegeți MySQL pentru a nu vă complica.

- **Doriți să instalați emulația MysqlPerl?** MysqlPerl este vechea interfață Perl pentru MySQL. Este depășită. Răspundeți negativ la această întrebare, cu excepția cazului când aveți scripturi MysqlPerl și doriți să activați suportul de emulație pentru acestea în modulul DBI.
- **Unde este instalat MySQL în sistemul dumneavoastră?** Acesta trebuie să fie „bunicul" catalogului care conține fișierele antet din MySQL, probabil `/usr/local` sau `/usr/local/mysql`, în afara cazului când ați instalat MySQL într-o locație non-standard.
- **Ce bază de date trebuie să folosesc pentru testarea driverelor MySQL?** Baza de date prestabilită este `test`, ceea ce nu ar trebui să ridice nici o problemă, decât dacă ați dezactivat accesul anonim la aceasta. În acest caz, trebuie să dați un nume de bază de date la care aveți acces și apoi să specificați un nume de utilizator și o parolă MySQL valabile pentru următoarele câteva întrebări.
- **Care este gazda pe care rulează baza de date?** Dacă rulați un server local, `localhost` trebuie să fie suficientă. Dacă nu, denumiți o gazdă care rulează un server la care aveți acces. Serverul MySQL trebuie să ruleze pe această gazdă atunci când rulați comanda `make test`; în caz contrar, testele vor eșua.
- **Care este numele de utilizator pentru conectarea la baza de date?**

  Care este parola pentru conectarea la baza de date?

  Numele și parola care se vor folosi pentru conectarea la serverul MySQL în vederea testării. Răspunsul prestabilit este `undef` pentru ambele întrebări. Astfel, driverul este obligat să se conecteze ca utilizator anonim. Dacă trebuie să vă conectați sub un anumit cont, specificați răspunsuri nevide.

Dacă aveți probleme la instalarea modulelor Perl, consultați fișierul `README` pentru a identifica distribuția relevantă, precum și arhivele cu mesaje pentru lista de corespondență aferentă DBI, deoarece acolo se pot găsi răspunsurile la majoritatea problemelor de instalare.

Dacă doriți să folosiți CGI.pm, poate că acesta se găsește deja în varianta dumneavoastră de instalare a limbajului Perl. Încercați să rulați comanda `perldoc CGI`. Dacă este afișată documentația CGI, atunci CGI este instalat. În caz contrar, procurați-vă distribuția, despachetați-o și instalați-o folosind aceleași instrucțiuni ca și cele pentru celelalte module Perl.

Dacă doriți să instalați modulul Apache `mod_perl` în vederea utilizării cu scripturile DBI bazate pe Web, vizitați regiunea `mod_perl` a sitului Web Apache, situată în locația următoare, pentru a obține mai multe informații:

```
http://perl.apache.org/
```

Instalarea limbajului Perl și a serverului Apache

Următoarele instrucțiuni ar trebui să-și facă efectul. Dacă întâmpinați probleme, verificați secțiunea „VERBOSE INSTALL" a fișierului `INSTALL` inclus în distribuția PHP. (Oricum, este indicat să citiți fișierul respectiv. Conține o mulțime de informații.)

Aceste instrucțiuni pornesc de la premisa că aveți la dispoziție o distribuție sursă a serverului Apache și că veți rula PHP ca modul Apache. De asemenea, se presupune că dispuneți de Apache 1.3.9 și de PHP 3.0.12, precum și că vă aflați într-un catalog în care se găsesc cataloagele `apache_1.3.9` și `php-3.0.12`, care conțin distribuțiile sursă. Dacă dispuneți de versiuni sau locații diferite, utilizați versiunile și căile de acces din sistemul dumneavoastră atunci când folosiți următoarele instrucțiuni.

Pentru a configura și construi PHP, începeți prin a rula următoarele comenzi (procedura arată puțin ciudat, deoarece începeți cu distribuția Apache, dar nu este nici o problemă):

```
% cd apache_1.3.9
% ./configure
```

Comanda `configure` pentru Apache este necesară deoarece unele dintre fișierele antet de care are nevoie PHP sunt generate la configurare. Dacă nu rulați `configure` în distribuția Apache, procesul de configurare a limbajului PHP nu va putea găsi fișierele. După ce ați configurat Apache, deplasați-vă în catalogul de distribuție PHP, configurați PHP, construiți-l și apoi instalați-l. De asemenea, puteți instala o copie a fișierului de inițializare PHP:

```
% cd ../php-3.0.12
% ./configure --with-mysql --with-apache=../apache_1.3.9 9
            --enable-track-vars
% make
% make install
% cp php3.ini-dist /usr/local/lib/php3.ini
```

Comanda `configure` pentru PHP îi indică acestuia că doriți suportul pentru MySQL și indică amplasamentul arborelui sursă Apache. Opțiunea `--enable-track-vars` activează conversia automată a datelor de intrare în formular în variabile care pot fi ușor accesibile din interiorul paginilor dumneavoastră PHP. Comenzile `make` compilează și instalează PHP. Pentru a instala fișierul de inițializare `php3.ini` aveți nevoie de contul `root`.

Când construiți PHP ca modul, „instalarea" acestuia înseamnă doar copierea modulului în arborele sursă Apache, astfel încât să poată fi legat în binarul Apache. După ce ați efectuat această operație, puteți compila și instala Apache:

```
% cd ../apache_1.3.9
% ./configure --activate-module=src/modules/php3/libphp3.a
% make
% make install
```

Aceste etape reconfigurează serverul Apache astfel încât să fie la curent cu fișierul modul PHP, apoi îl construiesc și îl instalează. Trebuie să editați fișierul de configurare Apache `httpd.conf` astfel încât Apache să recunoască scripturi PHP. Recunoașterea modulului PHP se bazează pe extensia numelor de fișier folosite pentru scripturile PHP. De exemplu, dacă doriți ca Apache să recunoască atât `.php`, cât și `.php3`, includeți următoarele linii în fișierul de configurare:

```
AddType application/x-httpd-php3 .php
AddType application/x-httpd-php3 .php3
```

Sufixele care se vor folosi depind de modul în care intenționați să vă denumiți scripturile PHP. Extensiile `.php` și `.php3` sunt, probabil, cele mai comune. O altă extensie comună

este .phtml. Le puteți pur și simplu activa pe toate, dacă doriți. Aceasta este oricum soluția cea mai indicată dacă urmează să instalați pagini pe care vi le procurați de la alte persoane, ce folosesc una sau alta dintre aceste extensii.

De asemenea, puteți cere serverului Apache să recunoască index.php sau index.php3 ca fișier prestabilit pentru un catalog, atunci când la sfârșitul unui URL nu este specificat nici un nume de fișier. Probabil că în fișierul de configurare veți găsi o linie care se prezintă astfel:

```
DirectoryIndex index.html
```

Înlocuiți-o cu următoarea:

```
DirectoryIndex index.html index.php index.php3
```

După ce ați editat fișierul de configurare Apache, opriți serverul httpd dacă acesta era deja în funcțiune și apoi porniți noul server httpd pe care tocmai l-ați instalat. Pe multe sisteme, următoarele comenzi (sub contul root) execută aceste operații:

```
# /usr/local/apache/bin/apachectl stop
# /usr/local/apache/bin/apachectl start
```

De asemenea, trebuie să configurați serverul Apache astfel încât să pornească, respectiv să se oprească la pornirea și oprirea sistemului. Vezi documentația Apache pentru instrucțiuni. În mod normal, aceasta implică rularea comenzii apachectl start la încărcarea sistemului de operare, respectiv apachectl stop la oprire.

Instalarea sistemului MySQL pe Windows

Puteți rula MySQL sub Windows 95, Windows 98 sau Windows NT. Pentru aceasta, este necesar ca suportul pentru TCP/IP să fie instalat, iar programul dumneavoastră Winsock trebuie să aibă cel puțin versiunea 2.

Sub Windows puteți instala două categorii de programe MySQL:

- Programe autonome, cum sunt cele pe care le instalați pentru UNIX (serverul mysqld și programe client, precum mysql și mysqladmin).
- MyODBC, driverul MySQL pentru ODBC, care permite altor programe (precum Access) să comunice cu serverele MySQL.

Distribuțiile Windows sunt toate disponibile din situl Web MySQL, sub formă de fișiere ZIP. Pentru a despacheta un asemenea fișier, pur și simplu executați dublu clic pe acesta. Dacă nu reușiți, folosiți un program precum Winzip sau pkunzip pentru despachetarea fișierelor. Distribuțiile primare sunt următoarele:

- mysqlwin-versiune.zip

  Distribuția completă (serverul și clienții).
- winclients-versiune.zip

  Numai programe client (mysql, mysqladmin, mysqldump și așa mai departe). Puteți folosi această distribuție dacă nu doriți să rulați serverul sub Windows.
- myodbc-versiune-win95.zip

  myodbc-versiune-nt.zip

  Suportul MyODBC pentru Windows 95 (sau Windows 98), respectiv pentru Windows NT.

- mysqlclient-versiune-cygwin-b20.tar.gz

  Clienți MySQL compilați cu setul de instrumente Cygnus. Acesta include mysqlc, o versiune a clientului mysql care dispune de o caracteristică în linie de comandă de editare a istoricului comenzilor. Dacă o instalați, va trebui să copiați fișierul bibliotecă cygwinb19.dll din C:\mysql\lib în catalogul de sistem Windows.

Instalarea unei distribuții client sau client-server

Dacă dispuneți de distribuția winclients-versiune.zip, care conține numai programe client, despachetați-o pentru a genera un dosar mysql. Mutați acest dosar în C:\.

Dacă dispuneți de distribuția mysqlwin-versiune.zip, care conține serverul și clienții, despachetați-o mai întâi pe aceasta. În dosarul rezultant, rulați programul Setup pentru a instala mysql în C:\mysql.

Puteți alege dintre următoarele servere:

Server	Descriere
mysqld	Server standard
mysqld-opt	Server optimizat pentru procesoarele Pentium
mysqld-nt	Server care poate fi instalat ca serviciu sub Windows NT

Puteți rula oricare dintre aceste servere sub NT, dar ultimul poate fi rulat numai sub NT.

Pentru a rula mysqld sau mysqld-opt, porniți serverul după cum urmează:

```
C:\> C:\mysql\bin\mysqld
C:\> C:\mysql\bin\mysqld-opt
```

Pentru a opri serverul, folosiți utilitarul mysqladmin:

```
C:\> C:\mysql\bin\mysqladmin -u root shutdown
```

Sub Windows NT, puteți rula serverul mysqld-nt ca serviciu:

```
C:\> C:\mysql\bin\mysqld-nt --install
```

Dacă porniți serverul mysqld-nt în acest mod, puteți specifica alte opțiuni inserându-le în fișierul cu opțiuni C:\my.cnf. Aceste opțiuni nu pot fi specificate în linia de comandă. Când mysqld-nt rulează ca serviciu, îl puteți porni sau opri folosind următoarele comenzi:

```
C:\> net start mysql
C:\> net stop mysql
```

De asemenea, puteți folosi utilitarul Services Control Manager, care se află în dosarul Control Panel, pentru a porni sau pentru a opri serverul, respectiv puteți face uz de utilitarul mysqladmin pentru a opri serverul de la linia de comandă:

```
C:\> C:\mysql\bin\mysqladmin -u root shutdown
```

Pentru a rula mysqld-nt ca program autonom, invocați-l astfel:

```
C:\> C:\mysql\bin\mysqld-nt --standalone
```

În acest caz, puteți specifica și alte opțiuni în linia de comandă după opțiunea --standalone, dacă doriți. Pentru a opri serverul, folosiți mysqladmin.

Dacă aveți probleme în a determina serverul să funcționeze, examinați notele referitoare la Windows din capitolul „Instalarea sistemului MySQL" din manualul de referință MySQL.

Instalarea prestabilită permite utilizatorului root din MySQL să se conecteze fără parolă. Pentru instrucțiuni privind stabilirea unei parole, consultați capitolul 11.

Instalarea programului MyODBC

Despachetați distribuția adecvată (există una pentru Windows 95 sau 98, respectiv o alta pentru Windows NT). În dosarul rezultant, rulați programul `Setup` pentru a instala driverul ODBC pentru MySQL. Va fi de asemenea instalat un panou de control ODBC, pe care îl puteți folosi pentru a configura driverul.

Dacă întâlniți vreo eroare, precum „Probleme la copierea fișierului MFC30.DLL", în timp ce instalați MyODBC, înseamnă că `MFC30.DLL` este folosit de o altă aplicație. În acest caz, încercați să reporniți sistemul Windows în modul sigur (safe mode) și rulați din nou programul `Setup`.

Când rulați panoul de control ODBC, veți vedea o fereastră care vă permite să stabiliți un nume al sursei de date (DSN). Executați clic pe rubrica User DSN și apoi executați clic pe butonul Add pentru a deschide o fereastră care conține driverele disponibile pentru sursele de date. Selectați driverul MySQL din listă și executați clic pe butonul Finish. Veți vedea o fereastră care vă permite să introduceți parametri de conexiune pentru sursa de date. Completați parametrii adecvați pentru conexiunea pe care doriți să o stabiliți și apoi executați clic pe butonul OK. De exemplu, pentru a configura o sursă de date pentru baza de date `samp_db` folosind serverul și contul de utilizator pe care le-am utilizat în cea mai mare parte a acestei cărți, voi completa câmpurile după cum urmează:

Numele câmpului	Valoarea câmpului
Windows DSN name (nume DSN pentru Windows)	`samp_db`
MySQL host (name or IP) (gazdă MySQL - nume sau IP)	`pit-viper.snake.net`
MySQL database name: (numele bazei de date MySQL)	`samp_db`
User (utilizator):	`paul`
Password (parolă):	`secret`

Acum trebuie să puteți folosi programe compatibile ODBC pentru a obține accesul la bazele de date MySQL.

Instalarea suportului pentru Perl DBI

Cel mai ușor lucru pe care îl aveți de făcut este să vă procurați distribuția Perl ActiveState din situl Web ActiveState (`http://www.activestate.com/`) și să o instalați. Apoi, preluați și instalați modulele Perl suplimentare de care aveți nevoie. Pentru aceasta se va folosi programul `ppm` (Perl Package Manager). Pentru a afla care sunt modulele deja instalate, folosiți această comandă:

```
C:\> C:\perl\bin\ppm info
```

Apoi, instalați modulele de care aveți nevoie folosind comenzile adecvate din următoarea listă. Probabil că CGI.pm este deja instalat, dar probabil că va trebui să instalați pachetele asociate modulului DBI:

```
C:\> C:\perl\bin\ppm install Data-Dumper
C:\> C:\perl\bin\ppm install DBI
C:\> C:\perl\bin\ppm install DBD-mysql
C:\> C:\perl\bin\ppm install CGI
```

Instalarea PHP și Apache

Sub Windows, PHP este un executabil separat, iar suportul pentru MySQL este un fișier DLL. Pentru instrucțiuni complete privind instalarea PHP pe Windows, cel mai bun lucru de făcut este să vizitați următoarea pagină Web:

```
http://www.umesd.k12.or.us/php/win32install.html
```

Obținerea și instalarea distribuției bazei de date demonstrative samp_db

Fișierele folosite pentru configurarea bazei de date demonstrative `samp_db`, precum și pentru accesul la aceasta, sunt disponibile la următoarea adresă (rețineți că adresa este sensibilă la diferența între majuscule și minuscule și trebuie tastată exact așa cum se vede mai jos):

```
http://mysql.he.net/Downloads/Contrib/Examples/
```

Regăsiți fișierul de distribuție `samp_db.tar.gz`, deplasați-vă în catalogul unde doriți să-l instalați și despachetați fișierul folosind una dintre aceste comenzi:

```
% tar zxf samp_db.tar.gz
% gunzip < samp_db.tar.gz | tar xf -
```

Când despachetați distribuția, aceasta va crea un catalog denumit `samp_db`, care conține mai multe elemente:

- Fișiere pentru crearea și încărcarea bazei de date `samp_db`. Acestea sunt folosite în capitolul 1, „Introducere în MySQL și SQL".
- Programele scrise în C din capitolul 6, „Interfața API MySQL pentru C".
- Scripturile Perl DBI și fișierul cu opțiuni `samp_db.cnf` din capitolul 7, „Interfața API pentru Perl DBI".
- Scripturile PHP și fișierul `include samp_db.inc` din capitolul 8, „Interfața API pentru PHP".

Catalogul `samp_db` conține un fișier `README` pe care trebuie să-l consultați pentru a primi instrucțiuni cu privire la utilizarea fișierelor de distribuție.

ANEXA B

Referință de tipuri de coloane

Această anexă descrie fiecare tip de coloană furnizat de MySQL. Mai multe informații despre utilizarea fiecărui tip sunt disponibile în capitolul 2, „Lucrul cu date în MySQL". Dacă nu se specifică altceva, tipurile enumerate au fost prezente în MySQL începând cel puțin de la versiunea MySQL 3.21.0.

Specificațiile privind numele tipurilor sunt scrise folosind următoarele convenții:

- **Paranteze drepte** ([ ]). Informații opționale.
- M. Lățimea maximă de afișare; dacă nu se specifică altfel, M trebuie să fie un întreg cuprins între 1 și 255.
- D. Numărul de cifre care urmează după virgulă, pentru tipurile cu parte fracționară; D trebuie să fie un întreg cuprins între 0 și 30. De asemenea, D nu trebuie să fie mai mare decât M – 2. În caz contrar, valoarea lui M este ajustată la D + 2.

În terminologia ODBC, M și D corespund termenilor de „precizie" și „scală".

Pentru fiecare descriere de tip, vor fi furnizate una sau mai multe dintre următoarele categorii de informații:

- **Semnificație.** O scurtă descriere a tipului.
- **Atribute permise.** Cuvinte cheie de descriere a atributelor opționale, care pot fi asociate cu tipul coloanei în instrucțiunile CREATE TABLE sau ALTER TABLE. Atributele sunt menționate în ordine alfabetică, dar aceasta nu corespunde în mod obligatoriu cu ordinea impusă de sintaxa instrucțiunilor CREATE TABLE sau ALTER TABLE. Pentru sintaxa acestor instrucțiuni, consultați Anexa D, „Referință de sintaxă SQL". Atributele prezentate în descrierea fiecărui tip de coloană se adaugă la atributele globale menționate mai jos.
- **Lungime permisă.** Lungimea maximă permisă a valorilor din coloane, pentru tipurile șir.
- **Interval.** Pentru tipurile numerice sau dată-oră, intervalul de valori pe care îl poate reprezenta tipul respectiv. Pentru tipuri numerice întregi, sunt date două intervale, deoarece coloanele de tip întreg pot fi cu semn sau fără semn, iar intervalele sunt diferite pentru fiecare caz în parte.
- **Valoare zero.** Pentru tipurile dată-oră, valoarea „zero" care este stocată dacă în coloană este inserată o valoare incorectă.
- **Valoare prestabilită.** Valoarea prestabilită, dacă în specificația tipului nu există nici un atribut DEFAULT explicit.
- **Spațiu de stocare necesar.** Numărul de octeți necesar pentru stocarea valorilor din tip. Pentru unele tipuri, această valoare este fixă. Numărul de octeți variază pentru alte tipuri, în funcție de lungimea valorii stocate în coloană.

- **Comparații.** Pentru tipurile șir, arată dacă comparațiile sunt sau nu sensibile la diferența între majuscule și minuscule. Se aplică și pentru sortare și indexare, deoarece aceste operații se bazează pe comparații.
- **Sinonime.** Eventuale sinonime ale numelui tipului.
- **Note.** Alte observații referitoare la tipul respectiv.
- **Atribute globale.** Unele atribute se aplică tuturor sau aproape tuturor tipurilor de coloane; acestea vor fi prezentate aici, nu în descrierea fiecărui tip. Pentru fiecare tip pot fi specificate atributele NULL sau NOT NULL. Atributul DEFAULT valoare_prestabilită poate fi specificat pentru toate tipurile, cu excepția tipurilor BLOB și TEXT.

Tipuri numerice

MySQL furnizează tipuri numerice pentru valori întregi și cu virgulă mobilă. Tipurile pot fi selectate în funcție de domeniul de valori pe care trebuie să îl reprezentați.

Pentru tipurile întregi, o coloană trebuie să fie PRIMARY KEY sau un index UNIQUE dacă este specificat un atribut AUTO_INCREMENT. Prin inserția valorii NULL într-o coloană AUTO_INCREMENT se include o valoare cu o unitate mai mare decât valoarea curentă maximă a coloanei. Dacă este specificat atributul UNSIGNED, nu sunt permise valori negative pentru tipurile întregi.

Dacă este specificat atributul ZEROFILL, valorile pentru tipurile numerice sunt completate cu zerouri inițiale (adică scrise în stânga primei cifre a valorii – N.T.) pentru a corespunde lățimii de afișare a coloanei.

TINYINT[(M)]

Semnificație. Un număr întreg de dimensiuni foarte mici

Atribute permise. AUTO_INCREMENT, UNSIGNED, ZEROFILL

Interval. Între -128 și 127 (între -2^7 și 2^7-1), sau între 0 și 255 (între 0 și 2^8-1) dacă are atributul UNSIGNED

Valoare prestabilită. NULL dacă în coloană se acceptă valoarea NULL, 0 în cazul în care coloana are atributul NOT NULL

Spațiu de stocare necesar. 1 octet

SMALLINT[(M)]

Semnificație. Un număr întreg de dimensiuni mici

Atribute permise. AUTO_INCREMENT, UNSIGNED, ZEROFILL

Interval. Între -32768 și 32767 (între -2^{15} și $2^{15}-1$), sau între 0 și 65535 (între 0 și $2^{16}-$ dacă are atributul UNSIGNED

Valoare prestabilită. NULL dacă în coloană se acceptă valoarea NULL, 0 în cazul în care coloana are atributul NOT NULL

Spațiu de stocare necesar. 2 octeți

MEDIUMINT[(*M*)]

Semnificație. Un număr întreg de dimensiuni medii

Atribute permise. AUTO_INCREMENT, UNSIGNED, ZEROFILL

Interval. Între –8388608 și 8388607 (între -2^{23} și 2^{23}-1), sau între 0 și 16777215 (între 0 și 2^{24}-1) dacă are atributul UNSIGNED

Valoare prestabilită. NULL dacă în coloană se acceptă valoarea NULL, 0 în cazul în care coloana are atributul NOT NULL

Spațiu de stocare necesar. 3 octeți

INT[(*M*)]

Semnificație. Un număr întreg de dimensiuni normale

Atribute permise. AUTO_INCREMENT, UNSIGNED, ZEROFILL

Interval. Între –2147483648 și 2147483647 (între -2^{31} și 2^{31}-1), sau între 0 și 4294967295 (între 0 și 2^{32}-1) dacă are atributul UNSIGNED

Valoare prestabilită. NULL dacă în coloană se acceptă valoarea NULL, 0 în cazul în care coloana are atributul NOT NULL

Spațiu de stocare necesar. 4 octeți

Sinonime. INTEGER[(M)]

BIGINT[(*M*)]

Semnificație. Un număr întreg de dimensiuni mari

Atribute permise. AUTO_INCREMENT, UNSIGNED, ZEROFILL

Interval. Între –9223372036854775808 și 9223372036854775807 (între -2^{63} și 2^{63}-1), sau între 0 și 18446744073709551615 (între 0 și 2^{64}-1) dacă are atributul UNSIGNED

Valoare prestabilită. NULL dacă în coloană se acceptă valoarea NULL, 0 în cazul în care coloana are atributul NOT NULL

Spațiu de stocare necesar. 8 octeți

FLOAT[(*M*,*D*)]

Semnificație. Un număr mic, cu virgulă mobilă, în simplă precizie (mai puțin precis decât DOUBLE)

Atribute permise. ZEROFILL

Interval. Valorile minime diferite de zero sunt ±1.175494351E-38; valorile maxime diferite de zero sunt ±3.402823466E+38

Valoare prestabilită. NULL dacă în coloană se acceptă valoarea NULL, 0 în cazul în care coloana are atributul NOT NULL

Spațiu de stocare necesar. 4 octeți

Sinonime. Anterior versiunii MySQL 3.23, FLOAT(4) este sinonim cu FLOAT cu valorile M și D prestabilite.

Notă. Începând cu MySQL 3.23, FLOAT(4) este un tip cu virgulă mobilă autentic (valorile sunt stocate la precizia integrală permisă de echipamentele dumneavoastră, nu sunt rotunjite la numărul prestabilit de cifre după virgulă).

DOUBLE[(*M*,*D*)]

Semnificație. Un număr mare, cu virgulă mobilă, în dublă precizie (mai precis decât FLOAT)

Atribute permise. ZEROFILL

Interval. Valorile minime diferite de zero sunt ±2.2250738585072014E-308, valorile maxime diferite de zero sunt ±1.7976931348623157E+308

Valoare prestabilită. NULL dacă în coloană se acceptă valoarea NULL, 0 în cazul în care coloana are atributul NOT NULL

Spațiu de stocare necesar. 8 octeți

Sinonime. DOUBLE PRECISION[(M,D)] și REAL[(M,D)] sunt sinonime cu DOUBLE[(M,D)]. Anterior versiunii MySQL 3.23, FLOAT(8) este sinonim cu DOUBLE cu valorile M și D prestabilite.

Notă. Începând cu MySQL 3.23, FLOAT(8) este un tip cu virgulă mobilă autentic (Valorile sunt stocate la precizia integrală permisă de echipamentele dumneavoastră, nu sunt rotunjite la numărul prestabilit de cifre după virgulă.)

DECIMAL(*M*,*D*)

Semnificație. Un număr cu virgulă mobilă, stocat sub forma unui șir (1 octet pentru fiecare cifră, virgulă sau caracter – (cratimă)).

Atribute permise. ZEROFILL

Interval. Intervalul maxim este același ca pentru DOUBLE: intervalul efectiv pentru un tip DECIMAL dat este determinat de M și D. Dacă D este 0, valorile coloanelor nu au cifre după virgulă sau parte fracționară.

Valoare prestabilită. NULL dacă în coloană se acceptă valoarea NULL, 0 în cazul în care coloana are atributul NOT NULL

Spațiu de stocare necesar. M octeți pentru versiunile sistemului MySQL anterioare versiunii 3.23, M + 2 octeți pentru versiunile MySQL 3.23 și ulterioare

Sinonime. NUMERIC(M,D)

Notă. Începând de la versiunea MySQL 3.23, valoarea lui M nu include octeții necesari pentru caracterul semn sau pentru punctul zecimal, în conformitate cu ANSI SQL.

Tipuri șir

Tipurile șir din MySQL sunt frecvent folosite pentru stocarea textelor, dar au un caracter general și pot conține date arbitrare. Tipurile șir pot conține valori de lungime maximă variabilă și pot fi selectate în funcție de modul în care doriți să se facă diferența între majuscule și minuscule.

CHAR (*M*)

Semnificație. Un șir de caractere de lungime fixă, unde lungimea se poate situa între 0 și M octeți, iar M trebuie să fie un întreg cuprins între 1 și 255, anterior versiunii MySQL 3.23, respectiv între 0 și 255, după versiunea MySQL 3.23. Șirurile mai scurte de M caractere sunt completate cu spații finale la stocare. La stocare, șirurile mai lungi de M caractere sunt amputate la lungimea M. Spațiile finale (adică inserate la dreapta ultimului caracter – N.T.) sunt eliminate la regăsirea valorilor.

Atribute permise. BINARY

Lungime permisă. Între 0 și M octeți

Valoare prestabilită. NULL dacă în coloană se acceptă valoarea NULL, "" în cazul în care coloana are atributul NOT NULL

Spațiu de stocare necesar. M octeți

Comparații. Insensibil la diferența între majuscule și minuscule (sensibil, dacă este specificat atributul BINARY)

VARCHAR (*M*)

Semnificație. Un șir de caractere de lungime variabilă, unde lungimea se poate situa între 0 și M octeți. M trebuie să fie un întreg cuprins între 1 și 255, respectiv între 0 și 255, începând de la versiunea MySQL 3.23. Spațiile finale din cadrul valorilor sunt eliminate la stocarea acestora. La stocare, șirurile mai lungi de M caractere sunt amputate la lungimea M.

Atribute permise. BINARY

Lungime permisă. Între 0 și M octeți

Valoare prestabilită. NULL dacă în coloană se acceptă valoarea NULL, "" în cazul în care coloana are atributul NOT NULL

Spațiu de stocare necesar. Lungimea valorii, plus un octet pentru înregistrarea lungimii

Comparații. Insensibil la diferența între majuscule și minuscule (sensibil, dacă este specificat atributul BINARY)

TINYBLOB

Semnificație. O valoare BLOB de mici dimensiuni

Atribute permise. Nici unul, în afara atributelor globale

Lungime permisă. Între 0 și 255 (între 0 și 2^8-1) octeți

Valoare prestabilită. NULL dacă în coloană se acceptă valoarea NULL, "" în cazul în care coloana are atributul NOT NULL

Spațiu de stocare necesar. Lungimea valorii, plus un octet pentru înregistrarea lungimii

Comparații. Sensibil la diferența între majuscule și minuscule

BLOB

Semnificație. O valoare BLOB de dimensiuni normale

Atribute permise. Nici unul, în afara atributelor globale

Lungime permisă. Între 0 și 65535 (între 0 și 2^{16}-1) octeți

Valoare prestabilită. NULL dacă în coloană se acceptă valoarea NULL, "" în cazul în care coloana are atributul NOT NULL

Spațiu de stocare necesar. Lungimea valorii, plus doi octeți pentru înregistrarea lungimii

Comparații. Sensibil la diferența între majuscule și minuscule

MEDIUMBLOB

Semnificație. O valoare BLOB de dimensiuni medii

Atribute permise. Nici unul, în afara atributelor globale

Lungime permisă. Între 0 și 16777215 (între 0 și 2^{24}-1) octeți

Valoare prestabilită. NULL dacă în coloană se acceptă valoarea NULL, "" în cazul în care coloana are atributul NOT NULL

Spațiu de stocare necesar. Lungimea valorii, plus trei octeți pentru înregistrarea lungimii

Comparații. Sensibil la diferența între majuscule și minuscule

LONGBLOB

Semnificație. O valoare BLOB de dimensiuni mari

Atribute permise. Nici unul, în afara atributelor globale

Lungime permisă. Între 0 și 4294967295 (între 0 și 2^{32}-1) octeți

Valoare prestabilită. NULL dacă în coloană se acceptă valoarea NULL, "" în cazul în care coloana are atributul NOT NULL

Spațiu de stocare necesar. Lungimea valorii, plus patru octeți pentru înregistrarea lungimii

Comparații. Sensibil la diferența între majuscule și minuscule

TINYTEXT

Semnificație. O valoare TEXT de mici dimensiuni

Atribute permise. Nici unul, în afara atributelor globale

Lungime permisă. Între 0 și 255 (între 0 și 2^8-1) octeți

Valoare prestabilită. NULL dacă în coloană se acceptă valoarea NULL, "" în cazul în care coloana are atributul NOT NULL

Spațiu de stocare necesar. Lungimea valorii, plus un octet pentru înregistrarea lungim

Comparații. Insensibil la diferența între majuscule și minuscule

TEXT

Semnificație. O valoare TEXT de dimensiuni normale

Atribute permise. Nici unul, în afara atributelor globale

Lungime permisă. Între 0 și 65535 (între 0 și 2^{16}-1) octeți

Valoare prestabilită. NULL dacă în coloană se acceptă valoarea NULL, "" în cazul în care coloana are atributul NOT NULL

Spațiu de stocare necesar. Lungimea valorii, plus doi octeți pentru înregistrarea lungimii

Comparații. Insensibil la diferența între majuscule și minuscule

MEDIUMTEXT

Semnificație. O valoare TEXT de dimensiuni medii

Atribute permise. Nici unul, în afara atributelor globale

Lungime permisă. Între 0 și 16777215 (între 0 și 2^{24}-1) octeți

Valoare prestabilită. NULL dacă în coloană se acceptă valoarea NULL, "" în cazul în care coloana are atributul NOT NULL

Spațiu de stocare necesar. Lungimea valorii, plus trei octeți pentru înregistrarea lungimii

Comparații. Insensibil la diferența între majuscule și minuscule

LONGTEXT

Semnificație. O valoare TEXT de dimensiuni mari

Atribute permise. Nici unul, în afara atributelor globale

Lungime permisă. Între 0 și 4294967295 (între 0 și 2^{32}-1) octeți

Valoare prestabilită. NULL dacă în coloană se acceptă valoarea NULL, "" în cazul în care coloana are atributul NOT NULL

Spațiu de stocare necesar. Lungimea valorii, plus patru octeți pentru înregistrarea lungimii

Comparații. Insensibil la diferența între majuscule și minuscule

ENUM("valoare1","valoare2",...)

Semnificație. O enumerare; valorile coloanelor pot avea exact una dintre valorile membrilor listei

Atribute permise. Nici unul, în afara atributelor globale

Valoare prestabilită. NULL, dacă în coloană se acceptă valoarea NULL, prima valoare din enumerare în cazul în care coloana are atributul NOT NULL

Spațiu de stocare necesar. 1 octet pentru enumerări cu 1-255 membri, 2 octeți pentru enumerări cu 256-65535 membri

Comparații. Insensibil la diferența între majuscule și minuscule (sensibil anterior versiunii MySQL 3.22.1)

SET("valoare1","valoare2",...)

Semnificație. Un set; valorile coloanelor pot avea zero sau mai multe dintre valorile membrilor listei

Atribute permise. Nici unul, în afara atributelor globale

Valoare prestabilită. NULL, dacă în coloană se acceptă valoarea NULL, "" (set vid) în cazul în care coloana are atributul NOT NULL

Spațiu de stocare necesar. 1 octet pentru seturi cu 1-8 membri, 2 octeți (seturi cu 9-16 membri), 3 octeți (seturi cu 17-24 membri), 4 octeți (seturi cu 25-32 membri) sau 8 octeți (seturi cu 33-64 membri)

Comparații. Insensibil la diferența între majuscule și minuscule (sensibil anterior versiunii MySQL 3.22.1)

Tipuri pentru dată și oră

MySQL furnizează tipuri pentru reprezentarea datelor de tip temporal în diferite forme. Există tipuri de dată și de oră, fie împreună, fie în combinație. Există un tip special „amprentă de timp" care este actualizat automat la modificarea unei înregistrări, respectiv un tip pentru stocarea anilor, dacă nu aveți nevoie de o dată completă.

DATE

Semnificație. O dată, în format "AAAA-LL-ZZ"

Atribute permise. Nici unul, în afara atributelor globale

Interval. Între "1000-01-01" și "9999-12-31"

Valoare zero. "0000-00-00"

Valoare prestabilită. NULL, dacă în coloană se acceptă valoarea NULL, "0000-00-00" în cazul în care coloana are atributul NOT NULL

Spațiu de stocare necesar. 3 octeți (4 octeți înainte de versiunea MySQL 3.22)

TIME

Semnificație. O oră, în format "hh:mm:ss" ("-hh:mm:ss" pentru valori negative) reprezintă timpul scurs, dar poate fi tratat ca oră din zi

Atribute permise. Nici unul, în afara atributelor globale

Interval. Între "-838:59:59" și "838:59:59"

Valoare zero. "00:00:00"

Valoare prestabilită. NULL, dacă în coloană se acceptă valoarea NULL, "00:00:00" în cazul în care coloana are atributul NOT NULL

Spațiu de stocare necesar. 3 octeți

Notă. Deși este folosită ca valoare zero atunci când sunt inserate valori incorecte într-o coloană `TIME`, `"00:00:00"` este o valoare corectă, care se încadrează în domeniul normal de valori.

DATETIME

Semnificație. O dată și o oră (ambele componente sunt obligatorii), în format `"AAAA-LL-ZZ hh:mm:ss"`

Atribute permise. Nici unul, în afara atributelor globale

Interval. Între `"1000-01-01 00:00:00"` și `"9999-12-31 23:59:59"`

Valoare zero. `"0000-00-00 00:00:00"`

Valoare prestabilită. `NULL`, dacă în coloană se acceptă valoarea `NULL`, `"0000-00-00 00:00:00"` în cazul în care coloana are atributul `NOT NULL`

Spațiu de stocare necesar. 8 octeți

TIMESTAMP[(*M*)]

Semnificație. O amprentă de timp (dată și oră), în format `"AAAALLZZhhmmss"`

Atribute permise. Nici unul, în afara atributelor globale

Interval. Între `19700101000000` și undeva în anul 2037

Valoare zero. `00000000000000`

Valoare prestabilită. Data și ora curentă. Rețineți că `DESCRIBE` și `SHOW COLUMNS` raportează valoarea prestabilită ca `NULL`

Spațiu de stocare necesar. 4 octeți

Notă. Prin inserția unei valori `NULL` în prima coloană `TIMESTAMP` a unui tabel se inserează data și ora curentă. Modificarea valorii oricărei alte coloane din rând determină actualizarea primei coloane `TIMESTAMP` cu data și ora modificării. Valorile sunt stocate și folosite intern în calcule la precizie completă cu 14 caractere, indiferent de lățimea de afișare. Dacă este specificat `NOT NULL` ca atribut, acesta va fi ignorat.

YEAR

Semnificație. Un an, în format `AAAA`

Atribute permise. Nici unul, în afara atributelor globale

Interval. Între `1900` și `2155`

Valoare zero. `0000`

Valoare prestabilită. `NULL`, dacă în coloană se acceptă valoarea `NULL`, `0000` în cazul în care coloana are atributul `NOT NULL`

Spațiu de stocare necesar. 1 octet

Notă. Tipul de coloană `YEAR` a fost introdus în MySQL 3.22.

ANEXA C

Referință de operatori și funcții

Această anexă enumeră toate funcțiile MySQL pe care le puteți folosi în instrucțiuni SQL. De asemenea, enumeră operatorii folosiți pentru construirea expresiilor. Evaluarea unei expresii implică deseori conversii de tip ale valorilor din expresia respectivă. Consultați capitolul 2, „Lucrul cu date în MySQL", pentru detalii privind circumstanțele în care se produce conversia de tip și regulile folosite de MySQL pentru conversia valorilor de la un tip la altul.

Dacă nu este specificat altceva, operatorii și funcțiile prezentate aici sunt prezente în MySQL începând cel puțin de la versiunea 3.21.0. Dacă un operator sau funcție se comportă într-un mod diferit față de cel prezentat aici, examinați notele de modificare din manualul de referință MySQL. Este foarte posibil ca versiunile mai vechi de MySQL să conțină hibe care au fost remediate deja.

Exemplele de operatori și funcții sunt scrise în formatul următor:

```
expresie                                    ⇨ rezultat
```

`expresie` ilustrează modul de utilizare a unui operator sau a unei funcții, iar `rezultat` indică rezultatul obținut din evaluarea expresiei. De exemplu:

```
LOWER ("ABC")                               ⇨ "abc"
```

Aceasta înseamnă că apelul la funcția `LOWER("ABC")` produce rezultatul șir `"abc"`.

Puteți verifica personal exemplele prezentate în această anexă, folosind programul `mysql`. Pentru a încerca exemplul precedent, invocați programul `mysql`, tastați expresia dată ca exemplu, precedată de `SELECT` și urmată de un caracter punct și virgulă, după care apăsați pe tasta Enter:

```
mysql> SELECT LOWER("ABC");
+----------+
| LOWER("ABC") |
+----------+
| abc      |
+----------+
```

MySQL nu impune ca o instrucțiune `SELECT` să aibă o clauză `FROM`, ceea ce facilitează exersarea utilizării operatorilor și a funcțiilor, introducând expresii arbitrare în acest mod. (Unele sisteme de baze de date nu vă permit să emiteți o instrucțiune `SELECT` fără clauza `FROM` – nefericită restricție!)

Exemplele includ instrucțiuni `SELECT` complete pentru funcții a căror utilizare nu poate fi demonstrată altfel. Secțiunea „Funcții de sumar" este scrisă în acest mod, deoarece funcțiile respective nu au sens decât dacă se face referire la un anumit tabel.

Numele funcțiilor, precum și operatorii de tip cuvânt, precum BETWEEN, pot fi specificate folosind atât majuscule, cât și minuscule.

Anumite tipuri de argumente ale funcțiilor apar în mod repetat și sunt reprezentate de nume cu o semnificație fixă:

- *expr* reprezintă o expresie; în funcție de context, aceasta poate fi o expresie numerică, șir, de tip dată sau oră, și poate încorpora constante, referințe la coloane de tabel sau alte expresii.
- *sir* reprezintă un șir; poate fi un șir literal, o referință la o coloană dintr-un tabel cu valori de tip șir sau o expresie care produce un șir.
- *n* reprezintă un întreg (ca și literele situate lângă n în alfabet).
- *x* reprezintă un număr cu virgulă mobilă (ca și literele plasate lângă x în alfabet).

Alte nume de argumente sunt folosite mai puțin frecvent și sunt definite în locul de utilizare. Componentele opționale ale secvențelor de operatori sau de apeluri la funcții sunt indicate prin paranteze drepte ([ ]).

Operatori

Operatorii sunt folosiți pentru a combina termenii din expresii pentru a executa operații aritmetice, pentru compararea valorilor, pentru a executa operații la nivel de bit sau logice, respectiv pentru a identifica echivalențele cu un model.

Precedența operatorilor

Operatorii au niveluri de precedență variabile. Nivelurile sunt prezentate în lista următoare, în ordine descrescătoare. Operatorii situați pe aceeași linie au aceeași precedență. Operatorii de la un nivel de precedență dat sunt evaluați de la stânga la dreapta. Operatorii de la un nivel de precedență mai ridicat sunt evaluați înaintea operatorilor de la un nivel de precedență mai redus.

```
BINARY
NOT !
- (minus unar)
* / %
+ -
<< >>
&
|
< <= = <=> != <> >= > IN IS LIKE REGEXP RLIKE
BETWEEN
AND &&
OR ||
```

Operatorii unari (minus unar, NOT și BINARY) se asociază mai strâns decât operatorii binari. Cu alte cuvinte, aceștia se grupează cu termenul imediat următor dintr-o expresie, nu cu restul expresiei în totalitatea sa.


```
-2+3                    ⇨ 1
-(2+3)                  ⇨ -5
```

Gruparea operatorilor

Parantezele rotunde de deschidere, respectiv închidere, se pot folosi pentru a grupa părți ale unei expresii, cu scopul de a anula precedența prestabilită a operatorilor, care determină ordinea în care sunt evaluați termenii unei expresii (vezi „Precedența operatorilor"). De asemenea, parantezele mai pot fi folosite pur și simplu pentru a asigura claritatea vizuală, pentru a facilita citirea unei expresii.

```
1 + 2 * 3 / 4           ⇨ 2.50
(((1 + 2) * 3) / 4)     ⇨ 2.25
```

Operatori aritmetici

Acești operatori execută operații aritmetice standard. Operatorii aritmetici lucrează cu numere, nu șiruri (deși șirurile care seamănă cu numerele sunt convertite automat la valoarea numerică adecvată). Operațiile aritmetice cu valori NULL produc rezultatul NULL.

- +

  Calculează suma argumentelor.

```
2 + 2                   ⇨ 4
3.2 + 4.7               ⇨ 7.9
"43bc" + "21d"          ⇨ 64
"abc" + "def"           ⇨ 0
```

  Exemplul final din această listă arată că operatorul de adunare nu servește ca operator de concatenare a șirurilor, ca în alte limbaje. În schimb, șirurile sunt convertite în numere înainte ca operația aritmetică să aibă loc. Șirurile care nu seamănă cu numere sunt convertite la 0. Pentru a concatena șirurile, folosiți funcția CONCAT().

- -

  Calculează diferența dintre operanzi, atunci când este folosit între doi termeni ai unei expresii. Determină negativul operandului, atunci când este folosit în fața unui singur termen (adică schimbă semnul termenului).

```
10 - 7                  ⇨ 3
- (10 - 7)              ⇨ -3
```

- *

  Calculează produsul operanzilor.

```
2 * 3                   ⇨ 6
2.3 * -4.5              ⇨ -10.3
```

- /

  Calculează câtul a doi operanzi. Împărțirea cu zero produce un rezultat NULL.

```
3 / 1                   ⇨ 3.00
1 / 3                   ⇨ 0.33
1 / 0                   ⇨ NULL
```

- %

  Operatorul modulo; calculează restul împărțirii lui m la n. m % n este similar cu MOD(m,n). Ca în cazul împărțirii, operatorul modulo cu un divizor egal cu zero returnează NULL.

```
12 % 4                                  ⇨ 0
12 % 5                                  ⇨ 2
12 % 0                                  ⇨ NULL
```

Pentru operatorii de adunare, scădere și înmulțire, operațiile aritmetice sunt executate cu valori BIGINT (întregi pe 64 de biți) dacă ambele argumente sunt întregi. Aceasta înseamnă că expresii cu valori mari pot depăși domeniul calculelor cu numere întregi pe 64 de biți, putând duce la rezultate imprevizibile:

```
9999999999999999 * 9999999999999999     ⇨ -7527149226598858751
99999999999 * 99999999999 * 99999999999 ⇨ -1504485813132150785
18014398509481984 * 18014398509481984   ⇨ 0
```

Pentru operatorii de împărțire și modulo, valorile BIGINT sunt folosite numai atunci când împărțirea se efectuează într-un context unde rezultatul este convertit la un întreg.

Operatori de comparație

Operatorii de comparație returnează 1 în cazul în care comparația este adevărată, respectiv 0 în caz contrar. Puteți compara numere sau șiruri. Operanzii sunt convertiți după necesități, în conformitate cu următoarele reguli:

- Cu excepția operatorului <=>, comparațiile care implică valori NULL determină rezultatul NULL. (Operatorul <=> este similar cu =, cu excepția faptului că NULL <=> NULL este o afirmație adevărată.)
- Dacă ambii operanzi sunt șiruri, atunci vor fi comparați din punct de vedere lexicografic, ca șiruri. Comparațiile între șiruri se execută folosind setul de caractere în vigoare pe server.
- Dacă ambii operanzi sunt numere întregi, atunci sunt comparați din punct de vedere numeric, ca întregi.
- Constantele hexazecimale care nu sunt comparate cu un număr sunt comparate sub formă de șiruri binare.
- Dacă unul din operanzi este o valoare TIMESTAMP sau DATETIME și celălalt este o constantă, operanzii sunt comparați sub formă de valori TIMESTAMP. Această operație are drept scop o mai bună comportare a comparațiilor pentru aplicațiile ODBC.
- Dacă nu se aplică nici una dintre regulile precedente, operanzii sunt comparați numeric, ca valori cu virgulă mobilă. Rețineți că aici este inclusă și situația în care se compară un șir cu un număr. Șirul este convertit într-un număr, ceea ce are ca rezultat o valoare zero dacă șirul nu seamănă cu un număr. De exemplu, "14.3" este convertit la 14.3, dar "L4.3" este convertit la zero.

Comparațiile următoare ilustrează regulile de mai sus:

```
2 < 12                                  ⇨ 1
"2" < "12"                              ⇨ 0
"2" < 12                                ⇨ 1
```


Prima comparație implică doi întregi, care sunt comparați ca numere. Cea de-a doua comparație implică două șiruri, care sunt comparate din punct de vedere lexicografic. Cea de-a treia comparație implică un șir și un număr, deci vor fi comparate ca valori cu virgulă mobilă.

Comparațiile între șiruri nu sunt sensibile la diferența între majuscule și minuscule decât în cazul în care comparația implică un șir binar. Astfel, se execută o comparație sensibilă la diferența între majuscule și minuscule dacă folosiți cuvântul cheie BINARY sau dacă executați comparații între valori din coloane CHAR BINARY, VARCHAR BINARY sau BLOB.

- =

  Are ca rezultat 1 dacă operanzii sunt egali, respectiv 0 în caz contrar.

```
1 = 1                                   ⇨ 1
1 = 2                                   ⇨ 0
"abc" = "abc"                           ⇨ 1
"abc" = "def"                           ⇨ 0
"abc" = "ABC"                           ⇨ 1
BINARY "abc" = "ABC"                    ⇨ 0
BINARY "abc" = "abc"                    ⇨ 1
"abc" = 0                               ⇨ 1
```

"abc" este egal atât cu "abc", cât și cu "ABC", deoarece în mod prestabilit comparațiile între șiruri nu sunt sensibile la diferența între majuscule și minuscule. Comparațiile dintre șiruri pot căpăta această sensibilitate dacă se folosește operatorul BINARY. "abc" este egal cu 0 deoarece este convertit la un număr, în conformitate cu regulile de comparație. Deoarece "abc" nu arată ca un număr, este convertit la 0.

- <=>

  Operator de egalitate care acceptă și valori NULL; este similar cu operatorul de egalitate, cu excepția faptului că determină valoarea 1 când operanzii sunt egali, chiar și atunci când au valoarea NULL.

```
1 <=> 1                                 ⇨ 1
1 <=> 2                                 ⇨ 0
NULL <=> NULL                           ⇨ 1
NULL = NULL                             ⇨ NULL
```

Ultimele două exemple prezintă modul diferit în care operatorii = și <=> tratează comparațiile cu operatori NULL.

Operatorul <=> a fost introdus în versiunea MySQL 3.23.0.

- != sau <>

  Determină rezultatul 1 dacă operanzii sunt inegali, respectiv 0 în caz contrar.

```
3.4 != 3.4                              ⇨ 0
"abc" <> "ABC"                          ⇨ 0
BINARY "abc" <> "ABC"                   ⇨ 1
"abc" != "def"                          ⇨ 1
```

- <

Determină rezultatul 1 dacă operandul din stânga este mai mic decât operandul din dreapta; în caz contrar, rezultatul este 0.

```
3 < 10                    ⇨ 1
105.4 < 10e+1             ⇨ 0
"abc" < "ABC"             ⇨ 0
"abc" < "def"             ⇨ 1
```

- <=

Determină rezultatul 1 dacă operandul din stânga este mai mic sau egal decât operandul din dreapta; în caz contrar, rezultatul este 0.

```
"abc" <= "a"              ⇨ 0
"a" <= "abc"              ⇨ 1
13.5 <= 14                ⇨ 1
(3 * 4) - (6 * 2) <= 0    ⇨ 1
```

- >=

Determină rezultatul 1 dacă operandul din stânga este mai mare sau egal decât operandul din dreapta; în caz contrar, rezultatul este 0.

```
"abc" >= "a"              ⇨ 1
"a" >= "abc"              ⇨ 0
13.5 >= 14                ⇨ 0
(3 * 4) - (6 * 2) >= 0    ⇨ 1
```

- >

Determină rezultatul 1 dacă operandul din stânga este mai mare decât operandul din dreapta; în caz contrar, rezultatul este 0.

```
PI() > 3                  ⇨ 1
"abc" > "a"               ⇨ 1
SIN(0) > COS(0)           ⇨ 0
```

- *expr* BETWEEN *min* AND *max*

Determină rezultatul 1 dacă `min` este mai mic sau egal cu `expr` și dacă `max` este mai mare sau egal cu `expr`. Dacă operanzii `expr`, `min` și `max` sunt toți de același tip, aceste expresii sunt echivalente:

```
expr BETWEEN min AND max
(min <= expr AND expr <= max)
```

Dacă operanzii nu sunt de același tip, se va produce o conversie de tip și este posibil ca aceste expresii să nu fie echivalente. `BETWEEN` este evaluat folosind comparații determinate în conformitate cu tipul operandului `expr`:

- Dacă *expr* este un șir, operanzii sunt comparați lexicografic, ca șiruri. Comparațiile sunt sau nu sensibile la diferența între majuscule și minuscule, dacă `expr` este sau nu un șir binar.
- Dacă *expr* este un întreg, operanzii sunt comparați numeric, ca întregi.
- Dacă nici una dintre regulile anterioare nu este adevărată, operanzii sunt comparați numeric, sub formă de numere cu virgulă mobilă.


```
"def" BETWEEN "abc" AND "ghi"     ⇨ 1
"def" BETWEEN "abc" AND "def"     ⇨ 1
13.3 BETWEEN 10 AND 20            ⇨ 1
13.3 BETWEEN 10 AND 13            ⇨ 0
2 BETWEEN 2 AND 2                 ⇨ 1
B BETWEEN "A" AND "a"             ⇨ 0
BINARY B BETWEEN "A" AND "a"      ⇨ 1
```

`BETWEEN` a fost introdus în MySQL versiunea 3.21.2.

- *expr* IN (*valoare1*,*valoare2*,...)

expr NOT IN (*valoare1*,*valoare2*,...)

`IN` determină rezultatul 1 dacă `expr` este una dintre valorile din listă, respectiv 0 în caz contrar. Pentru `NOT IN`, situația se inversează. Expresiile următoare sunt echivalente:

```
expr NOT IN (valoare1,valoare2,...)
NOT (expr IN (valoare1,valoare2,...))
```

Dacă toate valorile din listă sunt constante, MySQL le sortează și determină rezultatul testului `IN` folosind o căutare binară, care este foarte rapidă.

```
3 IN (1,2,3,4,5)                  ⇨ 1
"d" IN ("a","b","c","d","e")      ⇨ 1
"f" IN ("a","b","c","d","e")      ⇨ 0
3 NOT IN (1,2,3,4,5)              ⇨ 0
"d" NOT IN ("a","b","c","d","e")  ⇨ 0
"f" NOT IN ("a","b","c","d","e")  ⇨ 1
```

Operatorul `IN` a fost introdus în MySQL versiunea 3.21.0.

- *expr* IS NULL

expr IS NOT NULL

`IS NULL` determină rezultatul 1 dacă valoarea operandului `expr` este `NULL`, respectiv 0 în caz contrar. `IS NOT NULL` determină rezultatul invers. Expresiile următoare sunt echivalente:

```
expr IS NOT NULL
NOT (expr IS NULL)
```

`IS NULL` și `IS NOT NULL` trebuie utilizate pentru a determina dacă valoarea operandului `expr` este sau nu `NULL`. Pentru aceasta, nu puteți folosi operatorii de comparație obișnuiți, precum = și !=.

```
NULL IS NULL                      ⇨ 1
0 IS NULL                         ⇨ 0
NULL IS NOT NULL                  ⇨ 0
0 IS NOT NULL                     ⇨ 1
NOT (0 IS NULL)                   ⇨ 1
NOT (NULL IS NULL)                ⇨ 0
NOT NULL IS NULL                  ⇨ 1
```

Ultimul exemplu returnează rezultatul respectiv deoarece `NOT` se asociază mai strâns decât `IS` (vezi „Precedența operatorilor").

Operatori pe bit

Operațiile pe bit sunt efectuate folosind valori BIGINT (întregi pe 64 de biți), care limitează domeniul maxim al operațiilor. Operațiile pe bit care implică valorile NULL determină un rezultat NULL.

- |

  Evaluează rezultatul operației SAU (reuniune) la nivel de bit a operanzilor.

```
1 | 1                    ⇨ 1
1 | 2                    ⇨ 3
1 | 2| 4 | 8             ⇨ 15
1 | 2| 4 | 8 | 15        ⇨ 15
```

- &

  Evaluează rezultatul operației ȘI (intersecție) la nivel de bit a operanzilor.

```
1 & 1                    ⇨ 1
1 & 2                    ⇨ 0
7 & 5                    ⇨ 5
```

- <<

  Deplasează operandul stâng spre stânga, cu un număr de poziții de bit indicate de operandul din dreapta. Deplasarea cu o cantitate negativă determină rezultatul zero.

```
1 << 2                   ⇨ 4
2 << 2                   ⇨ 8
1 << 62                  ⇨ 4611686018427387904
1 << 63                  ⇨ -9223372036854775808
1 << 64                  ⇨ 0
```

  Ultimele două exemple ilustrează limitele calculelor pe 64 de biți.

  Operatorul << a fost introdus în MySQL versiunea 3.22.2.

- >>

  Deplasează operandul stâng spre dreapta, cu un număr de poziții de bit indicate de operandul din dreapta. Deplasarea cu o cantitate negativă determină rezultatul zero.

```
16 >> 3                  ⇨ 2
16 >> 4                  ⇨ 1
16 >> 5                  ⇨ 0
```

  Operatorul >> a fost introdus în MySQL versiunea 3.22.2.

Operatori logici

Operatorii logici (cunoscuți și sub numele de *operatori booleeni*, de la numele matematicianului George Boole, care a formalizat utilizarea acestora) testează adevărul sau falsitatea expresiilor. Toate operațiile logice returnează 1 pentru „adevărat", respectiv 0 pentru „fals". Operatorii logici interpretează operanzii diferiți de zero ca „adevărat", respectiv operanzii egali cu 0 ca „fals". Valorile NULL sunt manipulate conform indicațiilor aferente operatorilor.

Operatorii logici se așteaptă ca operanzii să fie numere, deci operanzii de tip șir sunt convertiți în numere înainte de evaluarea operatorului.

- NOT sau !

  Negație logică; determină rezultatul 1 dacă operandul următor este FALSE, respectiv 0 dacă operandul are valoarea TRUE, cu o singură excepție: NOT NULL este NULL.

```
NOT 0                    ⇨ 1
NOT 1                    ⇨ 0
NOT NULL                 ⇨ NULL
NOT 3                    ⇨ 0
NOT NOT 1                ⇨ 1
NOT "1"                  ⇨ 0
NOT "0"                  ⇨ 1
NOT ""                   ⇨ 1
NOT "abc"                ⇨ 1
```

- OR sau ||

  SAU logic; determină rezultatul 1 dacă oricare dintre operanzi este adevărat (diferit de zero sau NULL), respectiv 0 în caz contrar.

```
0 OR 0                   ⇨ 0
0 OR 3                   ⇨ 1
4 OR 2                   ⇨ 1
1 OR NULL                ⇨ 1
```

- AND sau &&

  ȘI logic; determină rezultatul 1 dacă ambii operanzi sunt adevărați (diferiți de zero sau NULL), respectiv 0 în caz contrar.

```
0 AND 0                  ⇨ 0
0 AND 3                  ⇨ 0
4 AND 2                  ⇨ 1
1 AND NULL               ⇨ 0
```

În MySQL, operatorii !, || și && indică operații logice, așa cum indică în C. Rețineți mai ales că operatorul || nu execută concatenarea șirurilor, așa cum procedează în unele versiuni ale sistemului SQL. Pentru concatenarea șirurilor, folosiți în schimb funcția CONCAT().

Operatori de conversie

BINARY determină tratarea operatorului imediat următor ca șir binar, astfel încât comparațiile care implică șirul devin sensibile la diferența între majuscule și minuscule. Dacă operandul următor este un număr, va fi convertit la șir:

```
"abc" = "ABC"            ⇨ 1
"abc" = BINARY "ABC"     ⇨ 0
BINARY "abc" = "ABC"     ⇨ 0
"2" < 12                 ⇨ 1
"2" < BINARY 12          ⇨ 0
```

BINARY determină o conversie de la număr la șir; apoi, comparația este executată după criterii lexicografice, deoarece ambii operanzi sunt șiruri.

BINARY a fost introdus în MySQL versiunea 3.23.0.

Operatori de determinare a echivalențelor cu un model

MySQL furnizează caracteristica de determinare a echivalențelor cu un model folosind LIKE și echivalența cu modele compuse din expresii regulate extinse folosind REGEXP. Echivalențele cu modelele SQL nu sunt sensibile la diferența între majuscule și minuscule, cu excepția situațiilor când șirul care trebuie găsit sau șirul mode sunt șiruri binare. Echivalențele cu modele compuse din expresii regulate extinse sunt întotdeauna sensibile la diferența între majuscule și minuscule.

Echivalența cu un model SQL reușește numai dacă modelul corespunde cu întregul șir care trebuie găsit. Echivalențele cu modele compuse din expresii regulate extinse reușesc dacă modelul se găsește oriunde în interiorul șirului.

- *sir* LIKE *mod* [ESCAPE 'c']

  sir NOT LIKE *mod* [ESCAPE 'c']

  LIKE execută o operație simplă SQL de stabilire a echivalențelor cu un model și determină valoarea 1, dacă șirul model mod corespunde întregii expresii șir sir. Dacă nu corespunde, rezultatul este 0. Pentru NOT LIKE, rezultatele se inversează. Aceste două expresii sunt echivalente:

  ```
  sir NOT LIKE mod [ESCAPE 'c']
  NOT (sir LIKE mod [ESCAPE 'c'])
  ```

  Dacă oricare dintre șiruri este NULL, rezultatul este NULL.

  În modelele SQL, două caractere au o semnificație specială și servesc drept caractere de înlocuire, așa cum se poate vedea mai jos:

Caracter	Semnificație
%	Corespunde oricărei secvențe de caractere alta decât NULL (inclusiv șirul vid)
_	Corespunde oricărui caracter individual

  Modelele pot conține unul sau ambele caractere de înlocuire:

  ```
  "catnip" LIKE "cat%"          ⇨ 1
  "dogwood" LIKE "%wood"        ⇨ 1
  "bird" LIKE "____"            ⇨ 1
  "bird" LIKE "___"             ⇨ 0
  "dogwood" LIKE "%wo__"        ⇨ 1
  ```

  Sensibilitatea la diferența între majuscule și minuscule a operației de stabilire a echivalențelor cu un model în SQL folosind operatorul LIKE este determinată de șirurile care se compară. În mod normal, comparațiile nu sunt sensibile la diferența între majuscule și minuscule. Dacă oricare dintre șiruri este un șir binar, comparația prezintă sensibilitatea respectivă:


  ```
  "abc" LIKE "ABC"              ⇨ 1
  BINARY "abc" LIKE "ABC"       ⇨ 0
  "abc" LIKE BINARY "ABC"       ⇨ 0
  ```

  Deoarece caracterul % corespunde oricărei secvențe de caractere, corespunde chiar și absenței oricărui caracter:

  ```
  "" LIKE "%"                   ⇨ 1
  "cat" LIKE "cat%"             ⇨ 1
  ```

  În MySQL, puteți folosi LIKE cu expresii numerice:

  ```
  50 + 50 LIKE "1%"             ⇨ 1
  200 LIKE "2__"                ⇨ 1
  ```

  Pentru a stabili o corespondență literală cu un caracter de înlocuire, dezactivați semnificația specială a acestuia precedându-l cu un caracter escape, și anume \:

  ```
  "100% pur" LIKE "100%"        ⇨ 1
  "100% pur" LIKE "100\%"       ⇨ 0
  "100% pur" LIKE "100\% pur"   ⇨ 1
  ```

  Dacă doriți să specificați un caracter escape altul decât \, specificați-l folosind o clauză ESCAPE:

  ```
  "100% pur" LIKE "100^%" ESCAPE '^'   ⇨ 0
  "100% pur" LIKE "100^% pur" ESCAPE '^'  ⇨ 1
  ```

- *sir* REGEXP *mod*

  sir NOT REGEXP *mod*

  REGEXP stabilește echivalențe cu modele compuse din expresii regulate extinse. Expresiile regulate extinse sunt similare cu modelele folosite de utilitarele UNIX grep și sed. Secvențele de modele pe care le puteți folosi sunt prezentate în tabelul C.1.

  REGEXP determină un rezultat egal cu 1 dacă șirul model mod corespunde expresiei șir sir, respectiv 0 în caz contrar. NOT REGEXP determină rezultate inverse. Următoarele două expresii sunt echivalente:

  ```
  sir NOT REGEXP mod
  NOT (sir REGEXP mod)
  ```

  Dacă oricare dintre șiruri este NULL, rezultatul este NULL.

  Modelul nu trebuie să corespundă întregului șir; este suficient să se găsească undev în interiorul șirului.

  Construcțiile [...] și [^...] reprezintă clase de caractere. În cadrul unei clase, u domeniu de caractere se poate indica folosind o liniuță între cele două caracter extreme ale domeniului. De exemplu [a-z] corespunde oricărei litere minuscule, ia [0-9] corespunde oricărei cifre. Pentru a putea indica în cadrul unei clase un caracte literal de tip paranteză dreaptă de închidere, acesta trebuie să fie primul caracter al cla sei. Pentru a putea indica un caracter literal de tip cratimă, acesta trebuie să fie primu sau ultimul caracter al clasei. De asemenea, pentru a putea indica un caracter ^, acest nu trebuie să fie primul caracter după paranteza dreaptă de deschidere.

Tabelul C.1 Secvențe de expresii regulate extinse

Secvență	Semnificație
^	Corespunde începutului șirului
$	Corespunde sfârșitului șirului
.	Corespunde oricărui caracter individual, inclusiv caracterului linie nouă
[...]	Corespunde oricărui caracter care apare între paranteze
[^...]	Corespunde oricărui caracter care nu apare între paranteze
*e**	Corespunde unui număr de zero sau mai multe instanțe ale elementului de model *e*
e+	Corespunde unui număr de una sau mai multe instanțe ale elementului de model *e*
e?	Corespunde unui număr de 0-1 instanțe ale elementului de model *e*
e1|*e2*	Corespunde elementului de model *e1* sau *e2*
e{m}	Corespunde unui număr de *m* instanțe ale elementului de model *e*
e{m,}	Corespunde unui număr de *m* sau mai multe instanțe ale elementului de model *e*
e{,n}	Corespunde unui număr de instanțe ale elementului de model *e* cuprins între 0 și *n*
e{m,n}	Corespunde unui număr de instanțe ale elementului de model *e* cuprins între *m* și *n*
(...)	Grupează elementele de model într-un singur element
altele	Caracterele non-speciale corespund lor însăși

În cadrul unei clase de caractere se pot folosi și numeroase alte construcții speciale mai complicate, legate de colaționarea secvențelor și de clasele de echivalență. Pentru mai multe informații, examinați manualul de referință MySQL.

O echivalență cu o expresie regulată extinsă este stabilită dacă modelul se găsește oriunde în șirul căutat, dar puteți folosi caracterele ^ și $ pentru a forța găsirea modelului numai la începutul sau la sfârșitul șirului.

```
"abcde" REGEXP "b"        ⇨ 1
"abcde" REGEXP "^b"       ⇨ 0
"abcde" REGEXP "b$"       ⇨ 0
"abcde" REGEXP "^a"       ⇨ 1
"abcde" REGEXP "e$"       ⇨ 1
"abcde" REGEXP "^a.*e$"   ⇨ 1
```

Echivalențele cu expresiile regulate extinse sunt sensibile la diferența între majuscule și minuscule:

```
"abc" REGEXP "ABC"        ⇨ 0
"ABC" REGEXP "ABC"        ⇨ 1
```

- *sir* RLIKE *mod*

  sir NOT RLIKE *mod*

  RLIKE și NOT RLIKE sunt asemănătoare cu REGEXP, respectiv NOT REGEXP.

MySQL folosește o sintaxă similară cu C pentru secvențele escape din cadrul șirurilor. De exemplu \n, \t și \\ sunt interpretate drept caracter linie nouă, tabulator și backslash. Pentru a specifica asemenea caractere într-un model, dublați backslash-urile (\\n, \\t și \\\\). Un backslash este eliminat la analiza interogării, iar secvența escape rămasă este interpretată la identificarea echivalențelor cu modelul.

Funcții

Funcțiile sunt apelate pentru a efectua un calcul și pentru a returna o valoare. Funcțiile trebuie să fie invocate fără spații între numele funcției și paranteza care urmează după acesta:

```
NOW()        Corect
NOW ()       Incorect
```

Dacă transferați unei funcții mai multe argumente, argumentele trebuie separate prin virgulă. Este permisă inserția de spații în jurul argumentelor funcțiilor:

```
CONCAT("abc","def")        Corect
CONCAT( "abc" , "def" )    Și așa este corect
```

Funcții de comparație

- GREATEST(*expr1*,*expr2*,...)

  Returnează argumentul cel mai mare, unde noțiunea de „cel mai mare" este definită conform următoarelor reguli:

  - Dacă funcția este apelată într-un context de întregi sau toate argumentele sunt întregi, argumentele sunt comparate ca numere întregi.
  - Dacă funcția este apelată într-un context de valori cu virgulă mobilă sau toate argumentele sunt valori cu virgulă mobilă, argumentele sunt comparate ca valori cu virgulă mobilă.
  - În cazul în care cele două reguli precedente nu sunt adevărate, argumentele sunt comparate sub formă de șiruri. Comparațiile nu sunt sensibile la diferența între majuscule și minuscule decât dacă unul din argumente este un șir binar.

```
GREATEST(2,3,1)                    ⇨ 3
GREATEST(38.5,94.2,-1)             ⇨ 94.2
GREATEST("a","ab","abc")           ⇨ "abc"
GREATEST(1,3,5)                    ⇨ 5
GREATEST("A","b","C")              ⇨ "C"
GREATEST(BINARY "A","b","C")       ⇨ "b"
```

  Funcția GREATEST() a fost introdusă în versiunea 3.22.5. În versiunile anterioare, puteți folosi în schimb funcția MAX().

- IF(*expr1*,*expr2*,*expr3*)

  Dacă expr1 este adevărată (nu este 0 sau NULL), returnează expr2; în caz contrar, returnează expr3. IF() returnează un număr sau un șir, conform contextului în care este folosită funcția.

```
IF(1,"true","false")                ⇨ "true"
IF(0,"true","false")                ⇨ "false"
IF(NULL,"true","false")             ⇨ "false"
IF(1.3,"non-zero","zero")           ⇨ "non-zero"
IF(0.3,"non-zero","zero")           ⇨ "zero"
IF(0.3 != 0,"non-zero","zero")      ⇨ "non-zero"
```

expr este evaluată ca o valoare întreagă, iar ultimele trei exemple indică modul în care puteți fi păcălit de această comportare, dacă nu sunteți atent. 1.3 se convertește la valoarea întreagă 1, care este adevărată. Dar 0.3 se convertește la valoarea întreagă 0, care este falsă. Ultimul exemplu din listing prezintă modalitatea adecvată de utilizare a unui număr cu virgulă mobilă: testați-l folosind o comparație, întrucât comparația returnează 1 sau 0, în funcție de rezultatul testului.

- IFNULL(*expr1*,*expr2*)

Returnează *expr2* dacă valoarea expresiei *expr1* este NULL; în caz contrar, returnează expr1. IFNULL() returnează un număr sau un șir, conform contextului în care este folosită funcția.

```
IFNULL(NULL, "null")                ⇨ "null"
IFNULL("not null", "null")          ⇨ "not null"
```

- INTERVAL(*n*,*n1*,*n2*,...)

Returnează 0 dacă *n* < *n1*, 1 dacă *n* < *n2* și așa mai departe, respectiv –1 dacă *n* este NULL. Valorile *n1*, *n2*, ... trebuie să fie în ordine strict crescătoare (*n1* < *n2* < ...), deoarece se folosește o căutare binară rapidă. În caz contrar, funcția INTERVAL() va prezenta un comportament imprevizibil.

```
INTERVAL(1.1,0,1,2)                 ⇨ 2
INTERVAL(7,1,3,5,7,9)               ⇨ 4
```

- ISNULL(*expr*)

Returnează 1 dacă valoarea expresiei expr este NULL; în caz contrar, returnează 0.

```
ISNULL(NULL)                        ⇨ 1
ISNULL(0)                           ⇨ 0
ISNULL(1)                           ⇨ 0
```

- LEAST(*expr1*,*expr2*,...)

Returnează argumentul cel mai mic, unde noțiunea de „cel mai mic" este definită folosind aceleași reguli de comparație ca și cele pentru funcția GREATEST().

```
LEAST(2,3,1)                        ⇨ 1
LEAST(38.5,94.2,-1)                 ⇨ -1.0
LEAST("a","ab","abc")               ⇨ "a"
```

Funcția LEAST() a fost introdusă în MySQL 3.22.5. În versiunile anterioare, puteți folosi în schimb funcția MIN().

- STRCMP(*sir1*,*sir2*)

Această funcție returnează 1, 0 sau –1 dacă primul argument este din punct de vedere lexicografic mai mare, egal, respectiv mai mic decât al doilea. Dacă unul dintre argumente este NULL, funcția returnează NULL.


```
STRCMP("a","a")                     ⇨ 0
STRCMP("a","A")                     ⇨ 1
STRCMP("A","a")                     ⇨ -1
```

Funcții numerice

Funcțiile numerice returnează NULL dacă se produce o eroare. De exemplu, dacă transferați funcției argumente care nu se află în interiorul domeniului de valori sau care sunt incorecte din alt motiv, funcția va returna NULL.

- ABS(*x*)

Returnează valoarea absolută a lui *x*.

```
ABS(13.5)                           ⇨ 13.5
ABS(-13.5)                          ⇨ 13.5
```

- ACOS(*x*)

Returnează valoarea arccos(*x*), respectiv NULL dacă *x* nu se află în domeniul cuprins între –1 și 1.

```
ACOS(1)                             ⇨ 0.000000
ACOS(0)                             ⇨ 1.570796
ACOS(-1)                            ⇨ 3.141593
```

Funcția ACOS() a fost introdusă în versiunea MySQL 3.21.8.

- ASIN(*x*)

Returnează valoarea arcsin(*x*), respectiv NULL dacă *x* nu se află în domeniul cuprins între –1 și 1.

```
ACOS(1)                             ⇨ 1.570796
ACOS(0)                             ⇨ 0.000000
ACOS(-1)                            ⇨ -1.570796
```

Funcția ASIN() a fost introdusă în versiunea MySQL 3.21.8.

- ATAN(*x*)

Returnează valoarea arctangentei lui x.

```
ATAN(1)                             ⇨ 0.785398
ATAN(0)                             ⇨ 0.000000
ATAN(-1)                            ⇨ -0.785398
```

Funcția ATAN() a fost introdusă în MySQL 3.21.8.

- ATAN2(*x*,*y*)

Este similară cu ATAN(*x*,*y*), dar folosește semnele argumentelor pentru a determina cadranul valorii returnate (adică sfertul de cerc trigonometric în care se găsește aceasta – N.T.).

```
ATAN2(1,1)                          ⇨ 0.785398
ATAN2(1,-1)                         ⇨ 2.356194
ATAN2(-1,1)                         ⇨ -0.785398
ATAN2(-1,-1)                        ⇨ -2.356194
```

- CEILING(*x*)

Returnează cel mai mic întreg care nu este mai mic decât *x*.

```
CEILING(3.8)        ⇨ 4
CEILING(-3.8)       ⇨ -3
```

- COS(*x*)

Returnează cosinusul lui *x*, unde *x* este măsurat în radiani.

```
COS(0)              ⇨ 1.000000
COS((PI()/2))       ⇨ -1.000000
COS(PI()/2)         ⇨ 0.000000
```

Funcția COS() a fost introdusă în versiunea MySQL 3.21.8.

- COT(*x*)

Returnează cotangenta lui *x*, unde *x* este măsurat în radiani.

```
COT((PI()/2))       ⇨ 0.00000000
COT(PI()/4)         ⇨ 1.00000000
```

Funcția COT() a fost introdusă în versiunea MySQL 3.21.16.

- DEGREES(*x*)

Returnează valoarea lui *x*, convertită din radiani în grade.

```
DEGREES(PI())       ⇨ 180
DEGREES(PI()*2)     ⇨ 360
DEGREES(PI()/2)     ⇨ 90
DEGREES(-PI())      ⇨ -180
```

Funcția DEGREES a fost introdusă în MySQL 3.21.16.

- EXP(*x*)

Returnează valoarea e^x, unde *e* reprezintă baza logaritmilor naturali.

```
EXP(1)              ⇨ 2.718282
EXP(2)              ⇨ 7.389056
EXP(-1)             ⇨ 0.367879
1/EXP(1)            ⇨ 0.36787944
```

- FLOOR(*x*)

Returnează cel mai mare întreg care nu este mai mare decât *x*.

```
FLOOR(3.8)          ⇨ 3
FLOOR(-3.8)         ⇨ -4
```

- LOG(*x*)

Returnează logaritmul natural (în bază *e*) al lui *x*.

```
LOG(0)              ⇨ NULL
LOG(1)              ⇨ 0.000000
LOG(2)              ⇨ 0.693147
LOG(EXP(1))         ⇨ 1.000000
```

Puteți folosi funcția LOG() pentru a calcula logaritmul lui *x* pentru o bază arbitrară *b*; folosiți expresia LOG(*x*)/LOG(*b*).


```
LOG(100)/LOG(10)    ⇨ 2.00000000
LOG10(100)          ⇨ 2.000000
```

- LOG10(*x*)

Returnează logaritmul lui *x* în baza 10.

```
LOG10(0)            ⇨ NULL
LOG10(10)           ⇨ 1.000000
LOG10(100)          ⇨ 2.000000
```

- MOD(*m*,*n*)

Este identic cu *m* % *n*. Vezi „Operatori aritmetici".

- PI()

Returnează valoarea lui π.

```
PI()                ⇨ 3.141593
```

Funcția PI() a fost introdusă în versiunea MySQL 3.21.8.

- POW(*x*,*y*)

Returnează x^y, adică *x* ridicat la puterea *y*.

```
POW(2,3)            ⇨ 8.000000
POW(2,-3)           ⇨ 0.125000
POW(4,.5)           ⇨ 2.000000
POW(16,.25)         ⇨ 2.000000
```

- POWER(*x*,*y*)

Această funcție este sinonimă cu funcția POW(). A fost introdusă în MySQL 3.21.16.

- RADIANS(*x*,*y*)

Returnează valoarea lui *x*, convertită din grade în radiani.

```
RADIANS(0)          ⇨ 0
RADIANS(360)        ⇨ 6.28319
RADIANS(-360)       ⇨ -6.28319
```

Funcția RADIANS() a fost introdusă în MySQL 3.21.16.

- RAND()

RAND(*n*)

RAND() returnează o valoare aleatoare cu virgulă mobilă, situată în intervalul cuprins între 0.0 și 1.0. Funcția RAND(*n*) execută aceeași operație, folosind *n* ca valoare de amorsă pentru utilitarul de selectare aleatoare. Toate apelurile cu aceeași valoare a lui *n* returnează același rezultat; puteți folosi această proprietate atunci când aveți nevoie de o secvență repetabilă de numere.

```
RAND(10)            ⇨ 0.181091
RAND(10)            ⇨ 0.181091
RAND()              ⇨ 0.117195
RAND()              ⇨ 0.358596
RAND(10)            ⇨ 0.181091
```

În exemplele de mai sus, observați modul în care se comportă apelurile secvențiale la funcția RAND() atunci când funcția primește un argument, în comparație cu situațiile când acest argument nu este dat.

- ROUND(*x*)

  ROUND(*x*,*d*)

  Returnează valoarea lui *x*, rotunjită la un întreg. ROUND(*x*,*d*) returnează valoarea lui *x*, rotunjită la un număr cu d cifre după virgulă. Dacă d este 0, rezultatul nu are virgulă sau parte fracționară.

```
ROUND(15.3)                ⇨ 15
ROUND(15.5)                ⇨ 16
ROUND(-33.27834,2)         ⇨ -33.28
ROUND(1,4)                 ⇨ 1.0000
```

- SIGN(*x*)

  Returnează –1, 0 sau 1, dacă valoarea lui *x* este negativă, zero sau pozitivă.

```
SIGN(15.803)               ⇨ 1
SIGN(0)                    ⇨ 0
SIGN(-99)                  ⇨ -1
```

- SIN(*x*)

  Returnează valoarea sinusului lui *x*, unde *x* este măsurat în radiani.

```
SIN(0)                     ⇨ 0.000000
SIN(PI())                  ⇨ 0.000000
SIN(PI()/2)                ⇨ 1.000000
```

  Funcția SIN() a fost introdusă în versiunea MySQL 3.21.8.

- SQRT(*x*)

  Returnează rădăcina pătrată nenegativă a lui *x*.

```
SQRT(625)                  ⇨ 25.000000
SQRT(2.25)                 ⇨ 1.500000
SQRT(-1)                   ⇨ NULL
```

- TAN(*x*)

  Returnează valoarea tangentei lui x, unde x este măsurat în radiani.

```
TAN(0)                     ⇨ 0.000000
TAN((PI)/4)                ⇨ 1.000000
```

  Funcția TAN() a fost introdusă în versiunea MySQL 3.21.8.

- TRUNCATE(*x*,*d*)

  Returnează valoarea *x*, cu partea zecimală trunchiată la *d* cifre după virgulă. Dacă d este 0, rezultatul nu are virgulă sau parte zecimală. Dacă d este mai mare decât numărul de cifre după virgulă al argumentului *x*, partea zecimală este completată cu zerouri finale, până la dimensiunea dorită.

```
TRUNCATE(1.23,1)           ⇨ 1.2
TRUNCATE(1.23,0)           ⇨ 1
TRUNCATE(1.23,4)           ⇨ 1.2300
```

  Funcția TRUNCATE() a fost introdusă în versiunea MySQL 3.21.16.

Funcții șir

Majoritatea funcțiilor din această secțiune returnează un rezultat sub formă de șir. Unele din acestea, precum LENGTH(), preiau ca argumente șiruri și returnează un număr. Pentru funcțiile care operează cu șiruri și sunt bazate pe pozițiile șirurilor, poziția primului caracter (de la stânga la dreapta) este 1, nu 0.

- ASCII(*sir*)

  Returnează codul ASCII al primului caracter al șirului sir. Returnează 0 dacă sir este vid, respectiv NULL dacă *sir* este NULL.

```
ASCII("abc")               ⇨ 97
ASCII("")                  ⇨ 0
ASCII(NULL)                ⇨ NULL
```

  Funcția ASCII() a fost introdusă în MySQL 3.21.2.

- BIN(*n*)

  Returnează valoarea lui *n* în formă binară, ca șir. Următoarele două expresii sunt echivalente:

```
BIN(n)
CONV(n,10,2)
```

  Pentru mai multe informații, consultați descrierea funcției CONV().

  Funcția BIN() a fost introdusă în versiunea MySQL 3.22.4.

- CHAR(*n1*,*n2*,...)

  Interpretează argumentele sub formă de coduri ASCII și returnează un șir alcătuit din concatenarea valorilor caracterelor corespunzătoare. Argumentele NULL sunt ignorate.

```
CHAR(65)                   ⇨ "A"
CHAR(97)                   ⇨ "a"
CHAR(65,111,108,101,117,33) ⇨ "Aoleu!"
```

  Funcția CHAR() a fost introdusă în versiunea MySQL 3.21.0.

- CHARACTER_LENGTH(*sir*)

  Această funcție este un sinonim al funcției LENGTH().

- CHAR_LENGTH(*sir*)

  Această funcție este un sinonim al funcției LENGTH().

- COALESCE(*expr1*,*expr2*,...)

  Returnează primul element diferit de NULL din listă.

```
COALESCE(NULL,1/0,2,"a",45+97)  ⇨ "2"
```

  Această funcție a fost introdusă în MySQL versiunea 3.23.3.

- CONCAT(*sir1*,*sir2*,...)

  Returnează un șir alcătuit din concatenarea tuturor argumentelor sale. Returnează NULL dacă oricare dintre argumente este NULL. CONCAT() poate fi apelată cu mai multe argumente.

```
CONCAT("abc","def")                ⇨ "abcdef"
CONCAT("abc")                      ⇨ "abc"
CONCAT("abc", NULL)                ⇨ NULL
CONCAT("Salut",", ","pa")          ⇨ "Salut, pa"
```

- `CONV(n, din_baza, in_baza)`

Dacă se dă un număr *n* reprezentat în baza *din_baza*, funcția returnează o reprezentare șir a numărului n în baza *in_baza*. Rezultatul este `NULL` dacă orice argument este `NULL`. *din_baza* și *in_baza* trebuie să fie numere întregi, cuprinse între 2 și 36. *n* este tratat ca o valoare `BIGINT` (întreg pe 64 de biți), dar poate fi specificat sub formă de șir, deoarece numerele din baze mai mari decât 10 pot conține cifre non-zecimale. (Acesta este și motivul pentru care funcția `CONV()` returnează un șir; rezultatul poate conține caractere de la `A` la `Z` pentru bazele 11-36.) Rezultatul este `0` dacă `n` nu este un număr corect exprimat în baza `din_baza`. (De exemplu, dacă *din_baza* este 16 și *n* este de forma `"abcdefg"`, rezultatul este `0`, deoarece `g` nu este o cifră hexazecimală corectă.)

Caracterele non-zecimale din *n* pot fi specificate cu majuscule sau cu minuscule. Caracterele non-zecimale din rezultat vor fi scrise cu majuscule.

n este tratat, în mod prestabilit, ca număr fără semn. Dacă specificați valoarea *in_baza* ca număr negativ, `n` este tratat ca număr cu semn.

Conversia valorii 14, exprimată ca număr hexazecimal, în binar:

```
CONV("e",16,2)                     ⇨ "1110"
```

Conversia în octal a valorii 255 exprimate în binar:

```
CONV(11111111,2,8)                 ⇨ "377"
CONV("11111111",2,8)               ⇨ "377"
```

Funcția `CONV()` a fost introdusă în versiunea MySQL 3.22.4.

- `ELT(n,sir1,sir2,...)`

Returnează al *n*-lea șir din lista de șiruri *sir1,sir2,...*. Returnează `NULL` dacă nu există un șir la poziția respectivă sau dacă șirul respectiv sau *n* au valoarea `NULL`. Indexul primului șir este 1. Funcția `ELT()` este complementară cu funcția `FIELD()`.

```
ELT(3,"a","b","c","d","e")         ⇨ "c"
ELT(0,"a","b","c","d","e")         ⇨ NULL
ELT(6,"a","b","c","d","e")         ⇨ NULL
ELT(FIELD("b","a","b","c"),"a","b","c")  ⇨ "b"
```

- `EXPORT_SET(n,on,off,[separator,[contor_biti]])`

Returnează un șir alcătuit din șirurile `on` și `off`, separate prin șirul `separator`. `on` este folosit pentru fiecare bit cu valoarea 1 din `n`, iar `off` este folosit pentru fiecare bit cu valoarea `0`. `contor_biti` indică numărul maxim de biți din `n` care trebuie examinați. Șirul `separator` prestabilit este o virgulă, iar valoarea prestabilită a argumentului `contor_biti` este 64.

```
EXPORT_SET(7,"+","-","",5)         ⇨ "+++--"
EXPORT_SET(0xa,"1","0","",6)       ⇨ "010100"
EXPORT_SET(97,"Y","N",",",8)       ⇨ "Y,N,N,N,N,Y,Y,N"
```

Funcția `EXPORT_SET()` a fost introdusă în MySQL 3.23.2.

- `FIELD(sir,sir1,sir2,...)`

Găsește șirul `sir` în lista de șiruri *str1,str2...* și returnează indexul șirului care corespunde. Returnează zero dacă nu există nici un șir echivalent sau dacă `sir` este `NULL`. Indexul primului șir este 1. `FIELD()` este complementar cu `ELT()`.

```
FIELD("b","a","b","c")             ⇨ 2
FIELD("d","a","b","c")             ⇨ 0
FIELD(NULL,"a","b","c")            ⇨ 0
FIELD(ELT(2,"a","b","c"),"a","b","c")  ⇨ 2
```

- `FIND_IN_SET(sir,sir_lista)`

sir_lista este un șir alcătuit din sub-șiruri separate prin virgule (adică similar unei valori `SET`). Funcția `FIND_IN_SET()` returnează indexul șirului *sir* din cadrul listei *sir_lista*. Returnează `0` dacă `sir` nu există în *sir_lista*, respectiv `NULL` dacă oricare dintre argumente este `NULL`. Indexul primului sub-șir este 1.

```
FIND_IN_SET("vaca","elan,vaca,porc")   ⇨ 2
FIND_IN_SET("caine","elan,vaca,porc")  ⇨ 0
```

Funcția `FIND_IN_SET()` a fost introdusă în MySQL versiunea 3.21.22.

- `FORMAT(X,D)`

Formatează numărul *x* la un șir cu `d` cifre zecimale, în formatul `"nn,nnn.nnn"`[1]. Dacă `d` este `0`, rezultatul nu are nici un punct zecimal sau parte fracționară.

```
FORMAT(1234.56789,3)               ⇨ "1,234.568"
FORMAT(999999.99,2)                ⇨ "999,999.99"
FORMAT(999999.99,0)                ⇨ "1,000,000"
```

Observați rotunjirea prezentată în ultimul exemplu din lista anterioară.

- `HEX(n)`

Returnează valoarea lui *n* în format hexazecimal, sub formă de șir. Următoarele două expresii sunt echivalente:

```
HEX(n)
CONV(n,10,16)
```

Pentru mai multe informații, consultați descrierea funcției `CONV()`.

Funcția `HEX()` a fost introdusă în MySQL 3.22.4.

- `INSERT(sir,poz,lung,sir_nou)`

Returnează șirul *sir*, înlocuind sub-șirul care începe la poziția *poz* și are *lung* caractere lungime cu șirul *sir_nou*. Returnează șirul original dacă *poz* depășește domeniul, respectiv `NULL` dacă oricare dintre argumente este `NULL`.

```
INSERT("nighttime",6,4,"fall")     ⇨ "nightfall"
INSERT("sunshine",1,3,"rain or ")  ⇨ "rain or shine"
INSERT("sunshine",0,3,"rain or ")  ⇨ "sunshine"
```

- `INSTR(sir,subsir)`

`INSTR()` este similar formei cu două argumente a funcției `LOCATE()`, dar cu argumentele inversate. Următoarele două expresii sunt echivalente:

[1] În limba engleză, virgula desparte ordinele de mărime, iar punctul are rol zecimal (invers decât în limba română). – N.T.

INSTR(*sir,subsir*)
LOCATE(*subsir,sir*)

- LCASE(*sir*)

Returnează șirul *sir* cu toate caracterele scrise cu minuscule, respectiv NULL dacă *sir* este NULL.

```
LCASE("New York, NY")        ⇨ "new york, ny"
LCASE(NULL)                  ⇨ NULL
```

- LEFT(*sir,lung*)

Returnează primele *lung* caractere (numărate de la stânga la dreapta) ale șirului *sir*, respectiv întregul șir, dacă acesta nu dispune de un număr atât de mare de caractere. Dacă *sir* este NULL, funcția returnează NULL. Dacă argumentul lung este NULL sau mai mic decât 1, funcția returnează un șir vid.

```
LEFT("piciorul meu stang", 2)   ⇨ "pi"
LEFT(NULL,10)                   ⇨ NULL
LEFT("abc",NULL)                ⇨ ""
LEFT("abc",0)                   ⇨ ""
```

- LENGTH(*sir*)

Returnează lungimea șirului *sir*.

```
LENGTH("abc")                   ⇨ 3
LENGTH("")                      ⇨ 0
LENGTH(NULL)                    ⇨ NULL
```

- LOCATE(*subsir,sir*)

LOCATE(*subsir,sir,poz*)

Forma cu două argumente a funcției LOCATE() returnează poziția primei apariții a șirului *subsir* în cadrul șirului *sir*, respectiv 0 dacă *subsir* nu apare în cadrul șirului *sir*. Returnează NULL dacă orice argument este NULL. Dacă este dat argumentul de poziție *poz*, LOCATE() începe să caute *subsir* la poziția respectivă. Testul este sensibil la diferența între majuscule și minuscule.

```
LOCATE("b","abc")               ⇨ 2
LOCATE("b","ABC")               ⇨ 0
```

- LOWER(*sir*)

Această funcție este sinonimă cu LCASE().

- LPAD(*sir,lung,sir_comp*)

Returnează un șir alcătuit din valoarea șirului *sir*, completat la stânga cu șirul *sir_comp* pentru a se ajunge la un șir compus din *lung* caractere. Returnează *sir*, dacă acesta are deja *lung* caractere lungime.

```
LPAD("abc",12,"def")            ⇨ "defdefdefabc"
LPAD("abc",10,".")              ⇨ ".......abc"
LPAD("abc",2,".")               ⇨ "abc"
```

Funcția LPAD() a fost introdusă în versiunea MySQL 3.22.2.

- LTRIM(*sir*)

Returnează șirul *sir* fără spațiile de început, respectiv NULL dacă *sir* este NULL:


```
LTRIM("   abc   ")              ⇨ "abc   "
```

- MAKE_SET(*n,bit0_sir,bit1_sir,...*)

Construiește o valoare SET (un șir alcătuit din sub-șiruri separate prin virgule) bazată pe valoarea argumentului întreg *n* și pe șirurile *bit0_sir*, *bit1_sir*, ... Pentru fiecare bit activat din valoarea lui *n*, șirul corespunzător este inclus în rezultat. (Dacă bitul 0 este activat, rezultatul include *bit0_sir* etc.) Dacă *n* este 0, rezultatul este șirul vid. Dacă *n* este NULL, rezultatul este NULL. Dacă orice șir din listă este NULL, acesta este ignorat la construirea șirului rezultat:

```
MAKE_SET(8,"a","b","c","d","e")     ⇨ "d"
MAKE_SET(7,"a","b","c","d","e")     ⇨ "a,b,c"
MAKE_SET(2+16,"a","b","c","d","e")  ⇨ "b,e"
MAKE_SET(2|16,"a","b","c","d","e")  ⇨ "b,e"
MAKE_SET(-1,"a","b","c","d","e")    ⇨ "a,b,c,d,e"
```

Ultimul exemplu selectează toate șirurile, deoarece toți biții valorii –1 sunt activați.

Funcția MAKE_SET() a fost introdusă în versiunea MySQL 3.22.2.

- MID(*sir,poz,lung*)

Returnează un sub-șir al șirului *sir*, care începe la poziția *poz* și are lungimea de *lung* caractere. Dacă oricare dintre argumente este NULL, returnează NULL.

```
MID("ce exemplu plicticos",12,9)    ⇨ "plicticos"
```

- OCT(*n*)

Returnează valoarea lui *n* în formă octală, ca șir. Următoarele două expresii sunt echivalente:

OCT(*n*)
CONV(*n*,10,8)

Pentru mai multe informații, consultați descrierea funcției CONV().

Funcția OCT() a fost introdusă în versiunea MySQL 3.22.4.

- OCTET_LENGTH(*sir*)

Această funcție este un sinonim pentru LENGTH().

- POSITION(*subsir* IN *sir*) POSITION()

Această funcție este similară formei cu două argumente a funcției LOCATE(). Următoarele expresii sunt echivalente:

POSITION(*subsir* IN *sir*)
LOCATE(*subsir,sir*)

- REPEAT(*sir,n*)

Returnează un șir alcătuit din șirul *sir* repetat de *n* ori. Returnează un șir vid dacă *n* nu este pozitiv, respectiv NULL dacă oricare dintre argumente este NULL.

```
REPEAT("x",10)                  ⇨ "xxxxxxxxxx"
REPEAT("abc",3)                 ⇨ "abcabcabc"
```

- REPLACE(*sir,din_sir,in_sir*)

Returnează un șir alcătuit din șirul *sir*, unde toate aparițiile șirului *din_sir* înlocuite cu șirul *in_sir*. Returnează NULL dacă oricare dintre argumente este NULL. Dacă *to_sir* este vid, efectul constă în ștergerea tuturor aparițiilor șirului *din_sir*. Dacă *din_sir* este vid, funcția REPLACE() returnează șirul *sir* nemodificat.

```
REPLACE("abracadabra","a","oh")        ⇨ "ohbrohcohdohbroh"
REPLACE("abracadabra","a","")          ⇨ "brcdbr"
REPLACE("abracadabra","","oh")         ⇨ "abracadabra"
```

- `REVERSE(sir)`

Returnează un șir alcătuit din șirul `sir` cu caracterele scrise în ordine inversă. Returnează `NULL` dacă șirul `sir` este `NULL`.

```
REVERSE("abracadabra")                 ⇨ "arbadacarba"
REVERSE("Madam, I'm Adam")             ⇨ "madA m'I ,madaM"
```

Funcția `REVERSE()` a fost introdusă în versiunea MySQL 3.21.19.

- `RIGHT(sir,lung)`

Returnează primele *lung* caractere ale șirului *sir*, numărate de la dreapta la stânga, respectiv întregul șir, dacă acesta nu conține un număr atât de mare de caractere. Returnează `NULL` dacă șirul este `NULL`. Returnează șirul vid dacă *lung* este `NULL` sau dacă este mai mic decât 1.

```
RIGHT("parasuta",4)                    ⇨ "suta"
```

- `RPAD(sir,lung,sir_compl)`

Returnează un șir alcătuit din valoarea șirului *sir*, completat la dreapta cu șirul *sir_comp* pentru a ajunge la un șir compus din *lung* caractere. Returnează *sir*, dacă acesta are deja *lung* caractere lungime.

```
LPAD("abc",12,"def")                   ⇨ "abcdefdefdef"
LPAD("abc",10,".")                     ⇨ "abc......."
LPAD("abc",2,".")                      ⇨ "abc"
```

Funcția `LPAD()` a fost introdusă în versiunea MySQL 3.22.2.

- `RTRIM(sir)`

Returnează șirul *sir* fără spațiile de sfârșit, respectiv `NULL` dacă *sir* este `NULL`.

```
LTRIM("   abc   ")                     ⇨ "   abc"
```

- `SOUNDEX(sir)`

Returnează un șir codificat cu algoritmul Soundex[2], calculat pornind de la șirul *sir*. Caracterele non-alfanumerice din *sir* sunt ignorate. Caracterele internaționale non-alfabetice din afara domeniului cuprins între `A` și `Z` sunt tratate ca vocale.

```
SOUNDEX("Cow")                         ⇨ "C000"
SOUNDEX("Cowl")                        ⇨ "C400"
SOUNDEX("Howl")                        ⇨ "H400"
SOUNDEX("Hello")                       ⇨ "H400"
```

- `SPACE(n)`

Returnează un șir alcătuit din *n* spații, respectiv șirul vid dacă *n* este non-pozitiv sau `NULL` dacă *n* este `NULL`.

```
SPACE(6)                               ⇨ "      "
```

[2] Pentru semnificația funcției Soundex și modul de utilizare a acesteia, vezi capitolul 13 al volumului *Visual Basic 6 – Baze de date*, apărut recent la Editura Teora. – N.T.


```
SPACE(0)                               ⇨ ""
SPACE(NULL)                            ⇨ NULL
```

Funcția `SPACE()` a fost introdusă în MySQL 3.21.16.

- `SUBSTRING(sir,poz)`

`SUBSTRING(sir,poz,lung)`

`SUBSTRING(sir FROM poz)`

`SUBSTRING(sir FROM poz FOR lung)`

Returnează un sub-șir al șirului *sir*, începând de la poziția *poz*. Dacă este dat un argument *lung*, returnează un sub-șir care are un număr de caractere egal cu `lung`; în caz contrar, returnează întreaga parte dreaptă a șirului *sir*, începând de la poziția `poz`.

```
SUBSTRING("abcdef",3)                  ⇨ "cdef"
SUBSTRING("abcdef",3,2)                ⇨ "cd"
```

Următoarele expresii sunt echivalente:

```
SUBSTRING(sir,poz,lung)
SUBSTRING(sir FROM poz FOR lung)
MID(sir,poz,lung)
```

- `SUBSTRING_INDEX(sir,delim,n)`

Returnează un sub-șir al șirului *sir*. Dacă *n* este pozitiv, `SUBSTRING_INDEX()` găsește cea de-a n-a apariție a șirului de delimitare *delim*, apoi returnează toate caracterele aflate la stânga acelui delimitator. Dacă `n` este negativ, `SUBSTRING_INDEX` găsește a n-a apariție a șirului *delim*, numărând de la capătul din dreapta al șirului *sir*, apoi returnează toate caracterele din dreapta delimitatorului respectiv. Dacă *delim* nu se află în *sir*, atunci va fi returnat întregul șir. Returnează `NULL` dacă orice argument are valoarea `NULL`.

```
SUBSTRING_INDEX("jar-jar","j",-2)      ⇨ "ar-jar"
SUBSTRING_INDEX(USER(),"@",1)          ⇨ "paul"
```

Funcția `SUBSTRING_INDEX()` a fost introdus în versiunea MySQL 3.21.15.

- `TRIM([[LEADING | TRAILING | BOTH] [trim_sir] FROM] sir)`

Returnează șirul `sir` fără instanțele inițiale și/sau finale ale șirului *trim_sir*. Dacă este specificată opțiunea `LEADING`, funcția `TRIM()` elimină aparițiile inițiale ale șirului *trim_sir*. Dacă este specificată opțiunea `TRAILING`, funcția `TRIM()` elimină aparițiile finale ale șirului *trim_sir*. Dacă este specificată opțiunea `BOTH`, funcția `TRIM()` elimină aparițiile inițiale și finale ale șirului *trim_sir*. Opțiunea prestabilită este `BOTH` dacă nu este specificată nici una dintre opțiunile `LEADING`, `TRAILING` sau `BOTH`. Dacă nu este specificat *trim_sir*, funcția `TRIM()` elimină spațiile.

```
TRIM("^" FROM "^^^XYZ^^")              ⇨ "XYZ"
TRIM(LEADING "^" FROM "^^^XYZ^^")      ⇨ "XYZ^^"
TRIM(TRAILING "^" FROM "^^^XYZ^^")     ⇨ "^^^XYZ"
TRIM(BOTH "^" FROM "^^^XYZ^^")         ⇨ "XYZ"
TRIM(BOTH FROM "   ABC   ")            ⇨ "ABC"
TRIM("   ABC   ")                      ⇨ "ABC"
```

Funcția `TRIM()` a fost introdusă în MySQL 3.21.12.

- UCASE(*sir*)

  Returnează șirul *sir* cu toate caracterele convertite la majuscule, respectiv NULL dacă *sir* este NULL.

  ```
  UCASE("New York, NY")      ⇨ "NEW YORK, NY"
  UCASE(NULL)                ⇨ NULL
  ```

- UPPER(*sir*)

  Această funcție este sinonimă cu UCASE().

Funcții de tip dată și oră

Funcțiile de tip dată și oră preiau diferite tipuri de argumente. În general, o funcție care presupune un argument DATE va accepta, de asemenea, un argument DATETIME sau TIMESTAMP și va ignora partea de oră a valorii.

- ADDATE(*data*,INTERVAL *expr interval*)

  Această funcție este sinonimă cu DATE_INTERVAL().

- CURDATE()

  Returnează data curentă sub formă de șir în formatul "AAAA-LL-ZZ", respectiv ca un număr AAAALLZZ, conform contextului în care este folosită funcția:

  ```
  CURDATE()                  ⇨ "1999-08-10"
  CURDATE() + 0              ⇨ 19990810
  ```

Valori valabile pentru dată și oră

Dacă nu furnizați valori valabile pentru dată și oră funcțiilor care acceptă atare argumente, nu vă așteptați la un rezultat rezonabil. Mai întâi verificați argumentele.

- CURRENT_DATE

  Această funcție este un sinonim pentru CURDATE(); observați lipsa parantezelor.

- CURRENT_TIME

  Această funcție este un sinonim pentru CURTIME(); observați lipsa parantezelor.

- CURRENT_TIMESTAMP

  Această funcție este sinonimă cu NOW(); observați lipsa parantezelor.

- CURTIME()

  Returnează ora curentă din zi sub formă de șir în formatul "hh:mm:ss", respectiv sub formă de număr în formatul hhmmss, conform contextului în care este folosită funcția.

  ```
  CURTIME()                  ⇨ "16:41:01"
  CURTIME() + 0              ⇨ 164101
  ```

  Funcția CURTIME() a fost introdusă în MySQL 3.21.12.

- DATE_ADD(*data*,INTERVAL *expr interval*)

  Preia o valoare de tip dată, respectiv dată-oră data, adaugă la aceasta un interval de timp și returnează rezultatul. expr specifică valoarea orei care trebuie adăugată la data (respectiv scăzută, dacă expr începe cu semnul minus), iar interval specifică modul

de interpretare a intervalului. Rezultatul este o valoare DATE dacă data este o valoare de tip DATE și dacă în calculul rezultatului nu sunt implicate valori referitoare la oră. Dacă această afirmație nu este adevărată, rezultatul este o valoare DATETIME. Rezultatul este NULL dacă data nu este o dată corectă.

```
DATE_ADD("1999-12-01",INTERVAL 1 YEAR)          ⇨ "2000-12-01"
DATE_ADD("1999-12-01",INTERVAL 60 DAY)          ⇨ "2000-01-30"
DATE_ADD("1999-12-01",INTERVAL -3 MONTH)        ⇨ "1999-09-01"
DATE_ADD("1999-12-01 08:30:00",INTERVAL 12 HOUR) ⇨ "1999-09-01
                                                   20:30:00"
```

Tabelul C.2 prezintă valorile permise pentru argumentul interval, semnificațiile lor și formatul în care trebuie specificate valorile pentru fiecare tip de interval. Cuvântul cheie INTERVAL și specificatorii interval pot fi menționați folosind atât majuscule, cât și minuscule.

Expresia expr care se adaugă la dată poate fi specificată sub formă de număr sau ca un șir, cu excepția situațiilor în care conține caractere altele decât cifrele, caz în care expresia trebuie specificată sub formă de șir. Caracterele de delimitare sunt arbitrare și pot fi alcătuite din orice semn de punctuație:

```
DATE_ADD("1999-12-01",INTERVAL "2:3" YEAR_MONTH) ⇨ "2002-03-01"
DATE_ADD("1999-12-01",INTERVAL "2-3" YEAR_MONTH) ⇨ "2002-03-01"
```

Componentele valorii expr sunt comparate de la dreapta la stânga cu părțile așteptate, în funcție de specificatorul interval. De exemplu, formatul așteptat pentru HOUR_SECOND este "hh:mm:ss". O valoare expr de "15:21" este interpretată sub forma "00:15:21", nu "15:21:00".

```
DATE_ADD("1999-12-01 12:00:00",INTERVAL "15:21" HOUR_SECOND)
                                          ⇨ "1999-12-01 12:15:21"
```

Tabelul C.2 Tipurile de interval pentru funcția DATE_ADD()

Tip	Semnificație	Formatul valorii
SECOND	Secunde	ss
MINUTE	Minute	mm
HOUR	Ore	hh
DAY	Zile	ZZ
MONTH	Luni	LL
YEAR	Ani	AA
MINUTE_SECOND	Minute și secunde	"mm:ss"
HOUR_MINUTE	Ore și minute	"hh:mm"
HOUR_SECOND	Ore, minute și secunde	"hh:mm:ss"
DAY_HOUR	Zile și ore	"ZZ hh"
DAY_MINUTE	Zile, ore și minute	"ZZ hh:mm"
DAY_SECOND	Zile, ore, minute și secunde	"ZZ hh:mm:ss"
YEAR_MONTH	Ani și luni	"AA-LL"

BIBLIOTECA

Dacă *interval* este YEAR, MONTH sau YEAR_MONTH, iar partea rezultatului corespunzătoare zilei este mai mare decât numărul de zile din luna rezultantă, ziua primește valoarea numărului maxim de zile din luna respectivă.

```
DATE_ADD("1999-12-31",INTERVAL 2 MONTH)    ⇨ "2000-02-29"
```

Funcția DATE_ADD() a fost introdusă în versiunea MySQL 3.22.4.

- DATE_FORMAT(*data*, *format*)

Formatează o valoare data de tip dată sau dată-oră în conformitate cu șirul de formatare format și returnează șirul rezultant. DATE_FORMAT() se poate folosi pentru a reformata valorile DATE sau DATETIME din forma folosită de MySQL cu scopul de a furniza orice format doriți.

```
DATE_FORMAT("1999-12-01","%M %e, %Y")      ⇨ "December 1, 1999"
DATE_FORMAT ("1999-12-01","The %D of %M") ⇨ "The 1st of December"
```

Tabelul C.3 prezintă specificatorii disponibili care se pot folosi în șirul de formatare.

Începând de la versiunea MySQL 3.23.0, este necesară utilizarea caracterului % anterior codului de formatare. În versiunile anterioare, acest caracter este permis, dar are un caracter opțional. Caracterele prezente în șirul de formatare care nu sunt prezente în tabel sunt copiate literal în șirul rezultat.

Dacă faceți referire la specificatori de oră pentru o valoare DATE, partea de oră a valorii este tratată ca "00:00:00".

```
DATE_FORMAT("1999-12-01","%i")             ⇨ "00"
```

Funcția DATE_FORMAT() a fost introdusă în versiunea MySQL 3.21.14.

Tabelul C.3 Specificatori de formatare pentru funcția DATE_FORMAT()

Specificator	Semnificație
%S, %s	Secundă în format numeric cu două cifre (00, 01, ..., 59)
%i	Minut în format numeric cu două cifre (00, 01, ..., 59)
%H	Oră în format numeric cu două cifre, 24 de ore (00, 01, ..., 23)
%h, %I	Oră în format numeric cu două cifre, 12 ore (01, 02, ..., 12)
%k	Oră în format numeric, 24 de ore (0, 1,... , 23)
%l	Oră în format numeric, 12 ore (1, 2,... , 12)
%T	Oră din zi în format 24 de ore (*hh:mm:ss*)
%r	Oră din zi, în format 12 ore (*hh:mm:ss* AM sau *hh:mm:ss* PM)
%p	AM sau PM
%W	Numele zilei din săptămână (Sunday, Monday, ..., Saturday)
%a	Numele abreviat al zilei din săptămână (Sun, Mon, ..., Sat[3])
%d	Ziua din lună, în format numeric cu două cifre (00, 01, ..., 31)
%e	Ziua din lună, în format numeric (1, 2, ..., 31)

[3] Numele zilelor din săptămână, respectiv ale lunilor din an, sunt returnate de program în limba engleză. – N.T.

Specificator	Semnificație
%D	Ziua din lună cu sufixele limbii engleze (1st, 2nd, 3rd, ...)
%w	Ziua din săptămână, în format numeric (0=duminică, 1=luni, ..., 6=sâmbătă)
%j	Ziua din an, în format numeric cu trei cifre (001, 002, ..., 366)
%U	Săptămâna (0, 1..., 52), unde duminica este prima zi a săptămânii
%u	Săptămâna (0, 1, ..., 52), unde lunea este prima zi a săptămânii
%M	Numele lunii (January, February, ..., December)
%b	Numele lunii, în formă abreviată (Jan, Feb, ..., Dec)
%m	Luna în format numeric cu două cifre (01, 02, ..., 12)
%c	Luna în format numeric (1, 2, ..., 12)
%Y	Anul, în format numeric cu patru cifre
%y	Anul, în format numeric cu două cifre
%%	Un caracter procent literal

- DATE_SUB(*data*,INTERVAL *expr interval*)

Execută operații aritmetice cu date în aceeași manieră ca și DATE_ADD(), cu excepția faptului că expr se scade din valoarea datei data. Pentru mai multe informații, consultați informațiile despre funcția DATE_ADD().

```
DATE_SUB("1999-12-01",INTERVAL 1 MONTH)    ⇨ "1999-11-01"
DATE_SUB("1999-12-01",INTERVAL "13-2" YEAR_MONTH)
                                           ⇨ "1986-10-01"
DATE_SUB("1999-12-01 04:53:12",INTERVAL "13-2" MINUTE_SECOND)
                                           ⇨ "1999-12-01 04:40:10"
DATE_SUB("1999-12-01 04:53:12",INTERVAL "13-2" HOUR_MINUTE)
                                           ⇨ "1999-12-01 15:51:12"
```

Funcția DATE_SUB() a fost introdusă în MySQL versiunea 3.22.4.

- DAYNAME(*data*)

Returnează un șir care conține numele zilei din săptămână (în limba engleză – N.T.) pentru valoarea de tip dată *data*.

```
DAYNAME("1999-12-01")                      ⇨ "Wednesday"
DAYNAME("1900-12-01")                      ⇨ "Saturday"
```

Funcția DAYNAME() a fost introdusă în MySQL versiunea 3.21.23.

- DAYOFMONTH(*data*)

Returnează valoarea numerică a zilei din lună corespunzătoare valorii de tip dată *data*, în domeniul cuprins între 1 și 31.

```
DAYOFMONTH("1999-12-01")                   ⇨ "1"
DAYOFMONTH("1999-12-25")                   ⇨ "25"
```

Funcția DAYOFMONTH() a fost introdusă în MySQL versiunea 3.21.22.

- DAYOFWEEK(*data*)

Returnează valoarea numerică a zilei din săptămână corespunzătoare valorii de tip dată *data*. Valorile zilelor din săptămână variază între 1 (duminică) și 7 (sâmbătă), conform standardului ODBC. Vezi și funcția WEEKDAY().

```
DAYOFWEEK("1999-12-05")      ⇨ 1
DAYNAME("1999-12-05")        ⇨ "Sunday"
DAYOFWEEK("1999-12-11")      ⇨ 7
DAYNAME("1999-12-05")        ⇨ "Saturday"
```

Funcția DAYOFWEEK() a fost introdusă în MySQL versiunea 3.21.15.

- DAYOFYEAR(*data*)

Returnează valoarea numerică a zilei din an aferente valorii de tip dată *data*, unde valoarea zilei din an este cuprinsă între 1 și 366.

```
DAYOFYEAR("1999-12-01")      ⇨ 335
DAYOFYEAR("2000-12-31")      ⇨ 366
```

- EXTRACT(*interval* FROM *dataora*)

Returnează partea din valoarea dată-oră *dataora* indicată de *interval*, care poate fi oricare dintre specificatorii de interval permiși pentru DATE_ADD().

```
EXTRACT(YEAR FROM "1999-12-01 13:42:19")        ⇨ 1999
EXTRACT(MONTH FROM "1999-12-01 13:42:19")       ⇨ 12
EXTRACT(DAY FROM "1999-12-01 13:42:19")         ⇨ 1
EXTRACT(HOUR_MINUTE FROM "1999-12-01 13:42:19") ⇨ 1342
EXTRACT(SECOND FROM "1999-12-01 13:42:19")      ⇨ 19
```

EXTRACT() poate fi de asemenea folosit cu valorile DATE; în acest caz, extragerea unei ore va determina rezultatul NULL.

Funcția EXTRACT() a fost introdusă începând cu MySQL versiunea 3.23.0.

- FROM_DAYS(*n*)

Dacă este dată o valoare numerică reprezentând numărul de zile scurse de la anul 0 (care se obține, în mod caracteristic, apelând funcția TO_DAYS()), returnează data corespunzătoare.

```
TO_DAYS("1999-12-01")        ⇨ 730454
FROM_DAYS(730454 + 3)        ⇨ "1999-12-04"
```

Funcția FROM_DAYS() este destinată numai pentru datele care se conformează calendarului gregorian (începând din anul 1582).

- FROM_UNIXTIME(*unix_timestamp*)

FROM_UNIXTIME(*unix_timestamp*,*format*)

Dacă este dată o valoare UNIX de tip amprentă de timp *unix_timestamp*, cum este cea returnată de funcția UNIX_TIMESTAMP(), funcția returnează o valoare de tip dată-oră în formatul "AAAA-LL-ZZ hh:mm:ss" sau un număr în format AAAALLZZhhmmss, conform contextului în care se folosește funcția. Dacă este dat argumentul *format*, valoarea returnată este formatată sub formă de șir, așa cum se procedează în cazul funcției DATE_FORMAT().


```
FROM_UNIXTIME(934340541)          ⇨ "1999-08-10 22:02:21"
FROM_UNIXTIME(944028000)          ⇨ "1999-12-01 00:00:00"
FROM_UNIXTIME(944028000, "%Y")    ⇨ "1999"
```

Funcția FROM_UNIXTIME() a fost introdusă în MySQL versiunea 3.21.5. Forma cu două argumente a fost introdusă în MySQL versiunea 3.21.8.

- HOUR(*ora*)

Returnează valoarea numerică a orei pentru valoarea de tip oră *ora*, valoare cuprinsă în intervalul 0-23.

```
HOUR("12:31:58")             ⇨ 12
```

Funcția HOUR() a fost introdusă în MySQL versiunea 3.21.22.

- MINUTE(*ora*)

Returnează valoarea numerică a minutelor pentru valoarea de tip oră *ora*, valoare cuprinsă în intervalul 0-59.

```
MINUTE("12:31:58")           ⇨ 31
```

Funcția MINUTE() a fost introdusă în MySQL versiunea 3.21.22.

- MONTH(*data*)

Returnează valoarea numerică a lunii din an aferente valorii de tip dată *data*, valoare cuprinsă în intervalul 1-12.

```
MONTH("1999-12-01")          ⇨ 12
```

Funcția MONTH() a fost introdusă în MySQL versiunea 3.21.22.

- MONTHNAME(*data*)

Returnează un șir conținând numele lunii (în limba engleză – N.T.) aferente valorii de tip dată *data*.

```
MONTHNAME("1999-12-01")      ⇨ "December"
```

Funcția MONTHNAME() a fost introdusă în MySQL versiunea 3.21.23.

- NOW()

Returnează data și ora curentă sub formă de șir, în formatul "AAAA-LL-ZZ hh:mm:ss", respectiv sub formă de număr în formatul AAAALLZZhhmmss, conform contextului în care este folosită funcția.

```
NOW()                        ⇨ "1999-08-10 18:51:43"
NOW() + 0                    ⇨ 19990810185143
```

- PERIOD_ADD(*perioada*,*n*)

Adaugă n luni la valoarea de tip perioadă *perioada* și returnează rezultatul. Formatul valorii returnate este AAAALL. Formatul argumentului perioada poate fi AAAALL sau AALL. Nici una nu este o valoare de tip dată.

```
PERIOD_ADD(199902,12)        ⇨ 200002
PERIOD_ADD(9902,-3)          ⇨ 199811
```

- PERIOD_DIFF(*perioada1*,*perioada2*)

Preia diferența între argumentele de tip perioadă și returnează numărul de luni cuprins între acestea. Argumentele pot avea formatul AAAALL sau AALL. Nici unul nu este o valoare de tip dată.

```
PERIOD_DIFF(200002,199902)        ⇨ 12
PERIOD_DIFF(199811,9902)          ⇨ -3
```

- QUARTER(*data*)

Returnează valoarea numerică a trimestrului din an aferent valorii de tip dată *data*, în domeniul cuprins între 1 și 4.

```
QUARTER("1999-12-01")             ⇨ 4
QUARTER("2001-01-01")             ⇨ 1
```

Funcția QUARTER()a fost introdusă începând de la MySQL versiunea 3.21.22.

- SECOND(*ora*)

Returnează valoarea numerică a numărului de secunde pentru valoarea de tip oră *ora*, în domeniul cuprins între 0-59.

```
SECOND("12:31:58")                ⇨ 58
SECOND(123158)                    ⇨ 58
```

Funcția SECOND() a fost introdusă în versiunea MySQL 3.21.22.

- SEC_TO_TIME(*secunde*)

Dacă este dat un număr de secunde *secunde*, returnează valoarea corespondentă a orei sub formă de șir în formatul "hh:mm:ss", respectiv sub formă de număr în formatul hhmmss, conform contextului în care este folosită funcția.

```
SEC_TO_TIME(29834)                ⇨ "08:17:14"
SEC_TO_TIME(29834) + 0            ⇨ 81714
```

Funcția SEC_TO_TIME() a fost introdusă în versiunea MySQL 3.21.5.

- SUBDATE(*data*,INTERVAL *expr interval*)

Această funcție este un sinonim pentru DATE_SUB().

- SYSDATE()

Această funcție este un sinonim pentru NOW().

- TIME_FORMAT(*ora*,*format*)

Formatează valoarea de tip oră *ora* în conformitate cu șirul de formatare *format* și returnează șirul rezultant. Șirul de formatare este similar celui folosit de către funcția DATE_FORMAT(), dar singurii specificatori care pot fi folosiți sunt cei referitori la oră. Utilizarea altor specificatori determină o valoare NULL sau 0.

```
TIME_FORMAT("12:31:58","%H %i")   ⇨ "12 31"
TIME_FORMAT(123158,"%H %i")       ⇨ "12 31"
```

Funcția TIME_FORMAT() a fost introdusă în MySQL versiunea 3.21.3.

- TIME_TO_SEC(*ora*)

Dacă este dată o valoare *ora* care reprezintă timpul scurs, returnează un număr care reprezintă numărul corespunzător de secunde.

```
TIME_TO_SEC("08:17:14")           ⇨ 29834
```

Funcția TIME_TO_SEC() a fost introdusă în MySQL versiunea 3.21.16.

- TO_DAYS(*data*)

Returnează o valoare numerică reprezentând valoarea de tip dată *data* convertită la numărul de zile scurs de la anul 0. Valoarea poate fi transferată funcției FROM_DAYS() pentru a o reconverti înapoi într-o dată.

```
TO_DAYS("1999-12-01")             ⇨ 730454
FROM_DAYS(730454 - 365)           ⇨ "1999-12-01"
```

Funcția TO_DAYS() este destinată numai datelor acoperite de calendarul gregorian (de la anul 1582).

- UNIX_TIMESTAMP()

UNIX_TIMESTAMP(*data*)

Când este apelată fără argumente, returnează numărul de secunde de la începutul epocii UNIX ("1970-01-01 00:00:00" GMT). Când este apelată cu un argument de tip dată *data*, returnează numărul de secunde între epocă și data respectivă. *data* poate fi specificat în mai multe moduri: sub formă de șir DATE sau DATETIME, sub formă de valoare TIMESTAMP sau sub formă de număr în formatul AAAALLZZ sau AALLZZ, ora locală.

```
UNIX_TIMESTAMP()                  ⇨ 934341073
UNIX_TIMESTAMP("1999-12-01")      ⇨ 944028000
UNIX_TIMESTAMP(991201)            ⇨ 944028000
```

- WEEK(*data*)

WEEK(*data*,*prima*)

Când este apelată cu un singur argument, returnează un număr care reprezintă săptămâna din an aferentă valorii de tip dată *data*, în domeniul cuprins între 0 și 52. Se presupune că duminica este prima zi a săptămânii. Când este apelat cu două argumente, returnează același tip de valoare, dar argumentul *prima* indică prima zi din săptămână. Dacă prima este 0, săptămâna începe duminică. Dacă *prima* este 1, săptămâna începe luni.

```
WEEK("1999-12-05")                ⇨ 49
WEEK("1999-12-05",0)              ⇨ 49
WEEK("1999-12-05",1)              ⇨ 48
```

Funcția WEEK() a fost introdusă în versiunea MySQL 3.21.22. Forma cu două argumente a fost introdusă în MySQL versiunea 3.22.1.

- WEEKDAY(*data*)

Returnează valoarea numerică a zilei din săptămână aferentă valorii de tip dată data. Valorile zilelor din săptămână sunt cuprinse între 0 (pentru luni) și 6 (pentru duminică); vezi și funcția DAYOFWEEK().

```
WEEKDAY("1999-12-05")             ⇨ 6
DAYNAME("1999-12-05")             ⇨ "Sunday"
WEEKDAY("1999-12-13")             ⇨ 0
DAYNAME("1999-12-05")             ⇨ "Monday"
```

- YEAR(data)

Returnează valoarea numerică a anului pentru valoarea de tip dată data, în domeniul cuprins între 1000 și 9999.

```
YEAR("1999-12-01")                ⇨ 1999
```

Funcția YEAR() a fost introdusă în versiunea MySQL 3.21.22.

Funcții de sumar

Funcțiile de sumar sunt de asemenea cunoscute sub numele de *funcții agregat*. Acestea calculează o valoare unică, pe baza unui grup de valori. Cu toate acestea, valoarea rezultantă este bazată numai pe valori diferite de NULL din rândurile selectate (cu excepția faptului că funcția COUNT(*) numără toate rândurile). Funcțiile de sumar se pot folosi pentru a rezuma coloane întregi, respectiv valori atunci când sunt grupate în funcție de valori distincte într-o altă coloană sau combinație de coloane. Vezi secțiunea „Generarea sumarelor" în capitolul 1, „Introducere în MySQL și SQL".

Pentru exemplele din această secțiune, vom presupune că aveți un tabel numit tabelul_meu, cu o coloană de valori întregi denumită coloana_mea, care conține șase rânduri, cu valorile 1, 3, 5, 7, 9 și NULL.

- AVG(*expr*)

  Returnează valoarea medie a argumentului *expr* pentru toate valorile diferite de NULL din rândurile selectate.

```
SELECT AVG(coloana_mea) FROM tabelul_meu        ⇨ 5.0000
SELECT AVG(coloana_mea)*2 FROM tabelul_meu      ⇨ 10.0000
SELECT AVG(coloana_mea*2) FROM tabelul_meu      ⇨ 10.0000
```

- BIT_AND(*expr*)

  Returnează valoarea ȘI la nivel de bit a argumentului *expr* pentru toate valorile diferite de NULL din rândurile selectate.

```
SELECT BIT_AND(coloana_mea) FROM tabelul_meu    ⇨ 1
```

  Funcția BIT_AND() a fost introdusă în MySQL versiunea 3.21.11.

- BIT_OR(*expr*)

  Returnează valoarea SAU la nivel de bit a argumentului *expr* pentru toate valorile diferite de NULL din rândurile selectate.

```
SELECT BIT_OR(coloana_mea) FROM tabelul_meu     ⇨ 15
```

  Funcția BIT_OR() a fost introdusă în MySQL versiunea 3.21.11.

- COUNT(*expr*)

  Cu orice argument diferit de *, returnează un număr al valorilor diferite de NULL din setul de rezultate. Cu argumentul *, returnează numărul tuturor rândurilor din setul de rezultate, indiferent dacă rândurile conțin sau nu valori NULL.

```
SELECT COUNT(coloana_mea) FROM tabelul_meu      ⇨ 5
SELECT COUNT(*) FROM tabelul_meu                ⇨ 5
```

  Funcția COUNT(*) fără clauza WHERE este optimizată astfel încât să returneze foarte rapid numărul de înregistrări din tabelul denumit în clauza FROM. Când sunt specificate mai multe tabele, COUNT(*) returnează produsul numerelor de rânduri din tabelele individuale:

```
SELECT COUNT(*) FROM tabelul_meu AS m1, tabelul_meu AS m2
                                                ⇨ 36
```


- MAX(*expr*)

  Returnează valoarea maximă a argumentului *expr* pentru toate valorile diferite de NULL din rândurile selectate. Funcția MAX() se mai poate folosi și cu șiruri, caz în care returnează valoarea cea mai mare din punct de vedere lexicografic.

```
SELECT MAX(coloana_mea) FROM tabelul_meu        ⇨ 9
```

- MIN(*expr*)

  Returnează valoarea minimă a argumentului *expr* pentru toate valorile diferite de NULL din rândurile selectate. Funcția MIN() se mai poate folosi și cu șiruri, caz în care returnează valoarea cea mai mică din punct de vedere lexicografic.

```
SELECT MIN(coloana_mea) FROM tabelul_meu        ⇨ 1
```

- STD(*expr*)

  Returnează abaterea standard a argumentului *expr* pentru toate valorile diferite de NULL din rândurile selectate.

```
SELECT STD(coloana_mea) FROM tabelul_meu        ⇨ 2.8284
```

- STDEV(*expr*)

  Această funcție este sinonimă cu STD().

- SUM(*expr*)

  Returnează suma argumentelor *expr* pentru toate valorile diferite de NULL din rândurile selectate.

```
SELECT SUM(coloana_mea) FROM tabelul_meu        ⇨ 25
```

Funcții diverse

Funcțiile din această secțiune nu se încadrează în nici una din categoriile anterioare.

- BENCHMARK(*contor*,*expr*)

  Evaluează în mod repetat expresia *expr* de *contor* ori. BENCHMARK() este o funcție oarecum neobișnuită, în sensul că este destinată utilizării în programul client mysql. Valoarea returnată este întotdeauna 0, deci nu este de nici un folos. Valoarea de interes o constituie timpul scurs între momentul afișării rezultatului interogării și momentul afișării valorii returnate:

```
mysql> SELECT BENCHMARK(100000,PASSWORD("secret"));
+-----------------------------------------+
| BENCHMARK(100000,PASSWORD("secret"))    |
+-----------------------------------------+
|                                       0 |
+-----------------------------------------+
1 row in set (1.63 sec)
```

  Timpul scurs reprezintă numai un indicator aproximativ al vitezei de evaluare a expresiei de către server, deoarece acesta reprezintă timpul măsurat „pe ceas" al clientului, nu timpul microprocesorului din server. Intervalul de timp poate fi influențat de factori precum încărcarea serverului, dacă serverul este într-o stare executabilă sau dacă este oprit la sosirea interogării etc. Va trebui să executați o interogare de mai multe ori, pentru a determina o valoare reprezentativă.

  Funcția BENCHMARK() a fost introdusă în MySQL versiunea 3.22.15.

- `BIT_COUNT(n)`

Returnează numărul de biți activați din argument, acesta fiind tratat ca o valoare `BIGINT` (întreg pe 64 de biți).

```
BIT_COUNT(0)      ⇨ 0
BIT_COUNT(1)      ⇨ 1
BIT_COUNT(2)      ⇨ 1
BIT_COUNT(7)      ⇨ 3
BIT_COUNT(-1)     ⇨ 64
```

- `DATABASE()`

Returnează un șir care conține numele bazei de date curente, respectiv un șir vid dacă nu există nici o bază de date curentă.

```
DATABASE()        ⇨ "samp_db"
```

- `DECODE(sir,parola)`

Dacă este dat un șir criptat *sir*, obținut ca rezultat al unui apel la funcția `DECODE()`, funcția îl decriptează folosind șirul parolă *parola* și returnează șirul rezultant.

```
DECODE(ENCODE("secret","amestec"),"amestec")   ⇨ "secret"
```

- `ENCODE(sir,parola)`

Criptează șirul *sir* folosind șirul parolă *parola* și returnează rezultatul sub formă de șir binar. Șirul poate fi decodificat folosind funcția `DECODE`. Deoarece rezultatul criptat este un șir binar, trebuie să folosiți o coloană care este de unul din tipurile `BLOB` dacă doriți să stocați șirul.

- `ENCRYPT(sir)`

`ENCRYPT(sir,salt)`

Criptează șirul *sir* și returnează șirul rezultant. Aceasta este o criptare ireversibilă. Argumentul *salt*, dacă este dat, trebuie să fie un șir format din două caractere. Dacă se specifică o valoare *salt*, rezultatul criptării șirului *sir* va fi de fiecare dată același. Dacă nu se specifică argumentul *salt*, apeluri identice la funcția `ENCRYPT()` vor determina în timp rezultate diferite.

```
ENCRYPT("secret","AB")    ⇨ "ABS5SGh1EL6bk"
ENCRYPT("secret","AB")    ⇨ "ABS5SGh1EL6bk"
ENCRYPT("secret")         ⇨ "z1oPSN18WRwFA"
ENCRYPT("secret")         ⇨ "12FJgqDtV0g7Q"
```

Funcția `ENCRYPT()` folosește apelul la funcția sistem `crypt()` din UNIX și depinde de aceasta în două moduri. Mai întâi, dacă funcția `crypt()` nu este disponibilă în sistemul dumneavoastră, `ENCRYPT()` va returna întotdeauna `NULL`. În al doilea rând, `ENCRYPT()` depinde de modul în care `crypt()` operează în sistemele unde este prezentă. În particular, pe unele sisteme `crypt()` examinează numai primele opt caractere ale șirului care va fi criptat.

Funcția `ENCRYPT()` a fost introdusă în versiunea MySQL 3.21.12. Începând de la versiunea MySQL 3.22.16, argumentul `salt` poate avea mai mult de două caractere lungime.

- `GET_LOCK(sir,expira)`

Funcția `GET_LOCK()` este folosită în conjuncție cu funcția `RELEASE_LOCK()` pentru a executa blocarea consultativă (cooperativă). Puteți folosi cele două funcții pentru a scrie aplicații care cooperează în funcție de starea unui nume de blocare convenit în prealabil.

`GET_LOCK()` este apelată cu un nume de blocare indicat de șirul `sir` și o valoare de expirare egală cu `expira` secunde. Funcția returnează 1 dacă blocarea a fost obținută cu succes în intervalul de expirare, 0 dacă încercarea de blocare a eșuat datorită expirării intervalului, respectiv `NULL` dacă s-a produs vreo eroare. Valoarea `expira` determină intervalul de tip alocat încercării de a obține o blocare, nu durata blocării. După ce a fost obținută, blocarea rămâne în vigoare până la anulare.

Apelul următor obține o blocare denumită `"Aneta"` și așteaptă maximum 10 secunde pentru aceasta:

```
GET_LOCK("Aneta", 10)
```

Blocarea se aplică numai șirului însuși. Nu blochează o bază de date, un tabel sau vreun rând sau o coloană dintr-un tabel. Cu alte cuvinte, blocarea nu împiedică nici un client să execute vreo operație cu tabelele din baza de date, motiv pentru care blocarea cu `GET_LOCK()` are numai un caracter consultativ – nu face decât să permită altor clienți care cooperează să determine dacă blocarea este sau nu în vigoare.

Un client care dispune de o blocare asupra unui nume blochează încercările altor clienți de a bloca numele (sau încercările executate de alte fire din cadrul unui client multi-fir care folosește mai multe conexiuni cu serverul). Să presupunem că clientul 1 blochează șirul `"Aneta"`. Dacă clientul 2 încearcă să blocheze același șir, va fi blocat până când clientul 1 anulează blocarea sau până la scurgerea perioadei de expirare. În primul caz, clientul 2 va reuși să preia blocarea; în al doilea caz, nu va reuși.

Deoarece doi clienți nu pot bloca un șir dat în același timp, aplicațiile care convin asupra unui nume pot folosi starea de blocare a numelui respectiv ca indicator privind oportunitatea execuției de operații relative la numele respectiv. De exemplu, puteți construi un nume de blocare bazat pe o valoare a unei chei unice a unui rând dintr-un tabel, pentru a permite blocarea cooperativă a rândului respectiv.

O blocare este anulată în mod explicit prin apelarea funcției `RELEASE_LOCK()` cu numele blocării:

```
RELEASE_LOCK("Aneta")
```

Funcția `RELEASE_LOCK()` returnează 1 dacă blocarea a fost eliberată cu succes, 0 dacă blocarea era deținută de o altă conexiune (vă puteți anula numai propriile dumneavoastră blocări), respectiv `NULL` dacă blocarea respectivă nu există.

Orice blocare deținută de o conexiune client este de asemenea anulată automat la terminarea conexiunii, respectiv dacă clientul emite un alt apel la funcția `GET_LOCK()` (o conexiune client poate bloca un singur șir la un moment dat). În ultimul caz, blocarea deținută este anulată înainte de obținerea noii blocări, chiar dacă numele blocării este același.

Funcția `GETLOCK(sir,0)` se poate folosi ca interogare ciclică pentru a determina, fără așteptare, dacă există sau nu o blocare a șirului `sir`. (Evident, șirul se va bloca în cazul în care nu este blocat, deci nu uitați să apelați funcția `RELEASE_LOCK()` dacă se impune acest lucru.)

Funcția `GET_LOCK()` a fost introdusă în MySQL versiunea 3.21.7.

- LAST_INSERT_ID()

  LAST_INSERT_ID(*expr*)

  Fără nici un argument, funcția returnează valoarea AUTO_INCREMENT cel mai recent generată în timpul sesiunii curente a serverului, respectiv 0 dacă nu a fost generată o asemenea valoare. Cu un argument, LAST_INSERT_ID() este destinată utilizării într-o instrucțiune UPDATE. Rezultatul este tratat în același mod ca și o valoare generată automat, care este utilă pentru generarea secvențelor.

  Mai multe detalii despre această funcție se pot găsi în capitolul 2, „Lucrul cu date în MySQL". Pentru ambele forme ale funcției LAST_INSERT_ID(), valoarea este menținută de server pentru fiecare conexiune în parte și nu va fi modificată de alți clienți, nici chiar de acei clienți care determină crearea de noi valori generate automat.

  Forma cu un singur argument a funcției LAST_INSERT_ID() a fost introdusă în versiunea MySQL 3.22.9.

- LOAD_FILE(*nume_fisier*)

  Citește fișierul *nume_fisier* și returnează conținutul sub formă de șir. Fișierul trebuie să fie localizat pe server, trebuie specificată calea sa de acces absolută (completă) și trebuie să fie universal accesibil la citire, pentru a vă asigura că nu încercați să citiți un fișier protejat.

  Deoarece fișierul trebuie să se găsească pe server, trebuie să aveți privilegiul FILE. Dacă una din aceste condiții nu este îndeplinită, funcția LOAD_FILE() returnează NULL.

  Funcția LOAD_FILE() a fost introdusă în versiunea MySQL 3.23.0.

- MD5(*sir*)

  Calculează o sumă de control a șirului *sir*, bazată pe algoritmul MD5 Message-Digest al firmei RSA Data Security, Inc. Valoarea returnată este un șir alcătuit din 32 de cifre hexazecimale.

  ```
  MD5("secret")          ⇨ "5ebe2294ecd0e0f08eab7690d2a6ee69"
  ```

  Funcția MD5() a fost introdusă în MySQL versiunea 3.23.2.

- PASSWORD(*sir*)

  Dacă este dat un șir *sir*, calculează și returnează un șir parolă criptat, de forma folosită în tabelele de acordare din MySQL. Aceasta este o criptare ireversibilă.

  ```
  PASSWORD("secret")     ⇨ "428567f408994404"
  ```

  Rețineți că funcția PASSWORD() *nu* folosește același algoritm ca și cel utilizat în UNIX pentru criptarea parolelor conturilor de utilizator. Pentru acel tip de criptare, folosiți funcția ENCRYPT().

- RELEASE_LOCK(*sir*)

  RELEASE_LOCK() este folosită în conjuncție cu GET_LOCK(). Pentru detalii, consultați descrierea funcției GET_LOCK.

  Funcția RELEASE_LOCK() a fost introdusă în versiunea MySQL 3.21.7.

- SESSION_USER()

  Această funcție este sinonimă cu USER().

- SYSTEM_USER()

  Această funcție este sinonimă cu USER().

- USER()

  Returnează un șir care reprezintă numele de utilizator al clientului curent. Începând de la versiunea MySQL 3.22.1, șirul este de forma "utilizator@gazda", unde utilizator este numele utilizatorului și gazda este numele gazdei de la care clientul s-conectat la server.

  ```
  USER()                          ⇨ "paul@localhost"
  SUBSTRING_INDEX(USER(),"@",1)   ⇨ "paul"
  SUBSTRING_INDEX(USER(),"@",-1)  ⇨ "localhost"
  ```

- VERSION()

  Returnează un șir care descrie versiunea serverului, de exemplu "3.22.25-log". Sufixel care pot apărea după numărul versiunii serverului sunt -log (este activată caracteristic de consemnare în jurnal), -debug (serverul rulează în mod depanare) sau -demo (serveru rulează în modul demo).

  ```
  VERSION()                       ⇨ "3.23.1-alpha-log"
  ```

  Funcția VERSION()a fost introdusă în versiunea MySQL 3.21.13.

ANEXA D

Referință de sintaxă SQL

Această anexă descrie fiecare dintre instrucțiunile SQL furnizate de MySQL. De asemenea, este prezentată sintaxa pentru scrierea comentariilor în programele SQL. Comentariile sunt folosite pentru redactarea textelor având caracter descriptiv care sunt ignorate de server și pentru a ascunde cuvinte cheie specifice sistemului MySQL (aceste cuvinte cheie vor fi executate de MySQL, dar vor fi ignorate de alte servere de baze de date). Vezi secțiunea „Sintaxa comentariilor".

Dezvoltarea sistemului MySQL este perpetuă, deci îmbunătățirile aduse implementării sale a sistemului SQL se produc sistematic. Este util să consultați ocazional manualul de referință MySQL pe suport electronic, pentru a fi la curent cu facilitățile adăugate. Manualul este disponibil la adresa `http://www.mysql.com/`.

Descrierile sintactice folosesc următoarele convenții:

- Informațiile opționale sunt incluse între paranteze drepte ([ ]). Dacă o listă este delimitată de paranteze drepte, se poate selecta o singură alternativă.
- Dacă o listă este delimitată prin paranteze acolade ({ }), se impune selectarea unei alternative.
- Barele verticale (|) separă articolele alternative dintr-o listă.
- Notația cu puncte de suspensie (...) arată că articolul anterior punctelor de suspensie se poate repeta.

Dacă nu se specifică altceva, instrucțiunile enumerate aici sunt prezente în MySQL cel puțin de la versiunea MySQL 3.22.0.

Instrucțiuni SQL

Această secțiune descrie sintaxa și semnificația fiecăreia dintre instrucțiunile SQL folosite în sistemul MySQL. O instrucțiune va eșua dacă nu dispuneți de privilegiile necesare pentru a le executa. De exemplu, `USE nume_baza_de_date` va eșua dacă nu aveți permisiunea de a obține accesul la baza de date `nume_baza_de_date`.

ALTER TABLE

```
ALTER [IGNORE] TABLE nume_tabel lista_actiuni
```

Instrucțiunea `ALTER TABLE` vă permite să redenumiți tabele sau să le modificați structura. Pentru a o folosi, specificați numele tabelului `nume_tabel`, apoi indicați specificațiile pentru una sau mai multe operații care trebuie efectuate asupra tabelului. Cuvântul cheie

`IGNORE` intră în acțiune dacă în cheile unice ale noului tabel rezultă valori duplicate. Fără `IGNORE`, efectul instrucțiunii `ALTER TABLE` este anulat. Cu `IGNORE`, rândurile care conțin valori duplicate pentru chei unice vor fi șterse.

Parametrul `lista_actiuni` specifică una sau mai multe acțiuni, separate prin virgulă. Fiecare acțiune este executată pe rând. O acțiune poate fi oricare din următoarele:

- `ADD [COLUMN] declaratie_coloana [FIRST | AFTER nume_coloana]`

  Adaugă o coloană la tabel. *`declaratie_coloana`* reprezintă declarația coloanei; este în același format ca și cel folosit pentru instrucțiunea `CREATE TABLE`. Coloana devine prima coloană din tabel, dacă este dat cuvântul cheie `FIRST`, respectiv este plasată după coloana specificată, dacă este dată opțiunea `AFTER nume_coloana`. Dacă amplasamentul coloanei nu este specificat, coloana devine ultima coloană din tabel.

```
ALTER TABLE membru
    ADD membru_id INT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY
ALTER TABLE membru
    ADD membru_id INT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY
    FIRST
ALTER TABLE membru
    ADD membru_id INT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY
    AFTER sufix
```

- `ADD INDEX [nume_index] (coloane_index)`

  Adaugă un index la tabel. Indexul se bazează pe coloanele specificate în *`coloane_index`*, care trebuie să fie toate coloane în tabelul *`nume_tabel`*. Dacă sunt precizate mai multe coloane, trebuie să le separați prin virgulă. Pentru coloane de tip `CHAR` și `VARCHAR`, puteți indexa un prefix al coloanei, folosind sintaxa `nume_coloana(n)` pentru a indexa primele n caractere ale valorilor coloanei. Pentru coloanele `BLOB` și `TEXT`, *trebuie* să specificați o valoare a prefixului; nu puteți indexa întreaga coloană. Dacă numele indexului *`nume_index`* nu este specificat, atunci se va selecta automat un nume, în funcție de numele primei coloane indexate.

- `ADD PRIMARY KEY (coloane_index)`

  Adaugă o cheie primară la coloanele date. Cheia primește numele `PRIMARY`. Opțiunea *`coloane_index`* este specificată ca în cazul instrucțiunii `ADD INDEX`. Dacă o cheie primară există deja, se va produce o eroare.

```
ALTER TABLE presedinte ADD PRIMARY KEY (nume, prenume)
```

- `ADD UNIQUE [nume_index] (coloane_index)`

  Adaugă un index cu valori unice la *`nume_tabel`*. Opțiunile *`nume_index`* și *`coloane_index`* sunt specificate ca în cazul instrucțiunii `ADD INDEX`.

```
ALTER TABLE absente ADD UNIQUE id_data (elev_id, data)
```

- `ALTER [COLUMN] nume_coloana {SET DEFAULT valoare | DROP DEFAULT}`

  Modifică valoarea prestabilită a coloanei date, fie înlocuind-o cu valoarea specificată, fie prin ștergerea valorii prestabilite curente. În ultimul caz, se va atribui o nouă valoare prestabilită, așa cum se va arăta în cazul instrucțiunii `CREATE TABLE`.

```
ALTER TABLE eveniment ALTER tip SET DEFAULT "C"
ALTER TABLE eveniment ALTER tip DROP DEFAULT
```

- CHANGE [COLUMN] *nume_coloana declaratie_coloana*

Modifică numele și definiția unei coloane. *nume_coloana* este numele curent al coloanei, iar *declaratie_coloana* este declarația coloanei în urma modificării. Opțiunea *declaratie_coloana* este în același format ca și cel folosit pentru instrucțiunea CREATE TABLE. Rețineți că declarația include noul nume al coloanei; dacă nu doriți să modificați numele, veți specifica același nume de două ori.

```
ALTER TABLE elev CHANGE nume nume VARCHAR(40)
ALTER TABLE elev CHANGE nume nume_elev CHAR(30) NOT NULL
```

- DROP [COLUMN] *nume_coloana*

Elimină coloana dată din tabel. În cazul în care coloana face parte din vreun index, atunci va fi eliminată din indexul sau indexurile respective. Dacă vor fi eliminate toate coloanele dintr-un index, indexul va fi de asemenea eliminat.

```
ALTER TABLE presedinte DROP sufix
```

- DROP INDEX *nume_index*

Elimină indexul dat din tabel.

```
ALTER TABLE membru DROP INDEX nume
```

- DROP PRIMARY KEY

Elimină cheia primară din tabel. Dacă un tabel nu are nici un index unic care a fost creat drept PRIMARY KEY, dar are unul sau mai multe indexuri UNIQUE, primul din aceste indexuri este șters.

```
ALTER TABLE presedinte DROP PRIMARY KEY
```

- MODIFY [COLUMN] *declaratie_coloana*

Modifică declarația unei coloane. Declarația de coloană *declaratie_coloana* este dată folosind același format ca și cel pentru descrierile de coloană din cazul instrucțiunii CREATE TABLE. Declarația începe cu un nume de coloană, unde se indică modul în care trebuie modificată coloana. Instrucțiunea MODIFY a fost introdusă în MySQL versiunea 3.22.16.

```
ALTER TABLE elev MODIFY nume VARCHAR(40) DEFAULT "" NOT NULL
```

- RENAME [AS] *nume_tabel_nou*

Modifică numele tabelului *nume_tabel* în *nume_tabel_nou*.

```
ALTER TABLE presedinte RENAME prezident
```

- *optiuni_tabel*

Specifică opțiunile pentru tabel din categoria care poate fi precizată în partea *optiuni_tabel* a unei instrucțiuni CREATE TABLE.

```
ALTER TABLE puncte TYPE = MYISAM CHECKSUM = 1
```

CREATE DATABASE

```
CREATE_DATABASE nume_baza_de_date
```

Creează o bază de date cu numele dat. Instrucțiunea eșuează dacă baza de date există deja sau dacă nu dispuneți de privilegiul adecvat pentru a o crea.

CREATE FUNCTION

```
CREATE [AGGREGATE] FUNCTION nume_functie
    RETURNS {STRING | REAL | INTEGER}
    SONAME nume_bibl_partajata
```

Specifică o funcție definită de utilizator (*user-defined function* - UDF) care va fi încărcată în tabelul func din baza de date mysql. *nume_functie* este numele sub care doriți să fie desemnată funcția în instrucțiunile SQL. Cuvântul cheie care urmează după RETURNS indică tipul valorii returnate de funcție. *nume_bibl_partajata* este calea de acces a fișierului care conține codul executabil sau funcția.

Cuvântul cheie AGGREGATE arată că funcția este o funcție de grupare, precum SUM() sau MAX(). AGGREGATE a fost introdus în versiunea MySQL 3.23.5.

CREATE FUNCTION impune ca serverul să fie construit ca binar cu legături dinamice (nu ca binar static), deoarece mecanismul UDF impune legarea dinamică. Pentru instrucțiuni referitoare la scrierea funcțiilor definite de utilizator, examinați manualul de referință MySQL.

CREATE INDEX

```
CREATE [UNIQUE] INDEX nume_index ON nume_tabel (coloane_index)
```

Adaugă un index cu numele *nume_index* la tabelul *nume_tabel*. Această instrucțiune este tratată ca și instrucțiunile ALTER TABLE ADD INDEX sau ALTER TABLE ADD UNIQUE, în conformitate cu absența sau prezența cuvântului cheie UNIQUE. Pentru detalii, examinați informațiile aferente instrucțiunii ALTER TABLE. Funcția CREATE INDEX a fost introdusă în versiunea MySQL 3.22.

CREATE TABLE

```
CREATE [TEMPORARY] TABLE [IF NOT EXISTS] nume_tabel
    (creare_definitie,...)
    [optiuni_tabel]
    [[IGNORE | REPLACE] instructiune_select]

creare_definitie:
    { declaratie_coloana
      | PRIMARY KEY (coloane_index)
      | KEY [nume_index] [coloane_index]
      | INDEX [nume_index] [coloane_index]
      | UNIQUE INDEX [nume_index] [coloane_index]
      | [CONSTRAINT simbol] FOREIGN KEY nume_index (coloane_index)
      | [definitie_referinta]
      | CHECK (expr) }
```

```
declaratie_coloana:
   nume_coloana tip_coloana
      [NOT NULL | NULL] [DEFAULT valoare_prestabilita]
      [AUTO_INCREMENT] [PRIMARY KEY] [definitie_referinta]

definitie_referinta:
   REFERENCES nume_tabel [(coloane_index)]
      [MATCH FULL | MATCH PARTIAL]
      [ON DELETE optiune_referinta]
      [ON UPDATE optiune_referinta]

optiune_referinta:
   {RESTRICT | CASCADE | SET NULL | NO ACTION | SET DEFAULT}
```

Instrucțiunea CREATE TABLE creează un tabel nou, denumit *nume_tabel*. Dacă este dat cuvântul cheie TEMPORARY, tabelul există numai până la terminarea, în condiții normale sau anormale, a conexiunii client curente, respectiv până la emiterea unei instrucțiuni DROP TABLE. Un tabel temporar este vizibil numai pentru clientul care l-a creat.

În mod normal, încercarea de a crea un tabel cu un nume care există deja va avea ca efect o eroare. Există două situații în care nu se vor produce erori. Mai întâi, dacă este specificată clauza IF NOT EXISTS, tabelul nu este creat, dar nu se produce nici o eroare. A doua situație: dacă este specificată opțiunea TEMPORARY și tabelul original nu este un tabel temporar, atunci va fi creat noul tabel temporar, dar tabelul original denumit *nume_tabel* devine ascuns clientului pe durata existenței tabelului temporar. Tabelul original rămâne vizibil pentru ceilalți clienți. Tabelul original redevine vizibil din nou fie la următoarea sesiune client, fie dacă este emisă o instrucțiune DROP TABLE explicită pentru tabelul temporar.

Lista *creare_definitie* denumește coloanele și indexurile pe care doriți să le creați. Lista este opțională dacă creați tabelul prin intermediul unei instrucțiuni SELECT finale. Clauza *optiuni_tabel* vă permite să specificați diferite opțiuni pentru tabel. Dacă este specificată o instrucțiune finală *instructiune_select* (sub forma unei instrucțiuni SELECT arbitrare), tabelul este creat folosind setul de rezultate returnat de instrucțiunea SELECT. Aceste clauze vor fi descrise mai detaliat în secțiunile următoare.

Cuvântul cheie TEMPORARY, clauza IF NOT EXISTS, clauza optiuni_tabel și capacitatea de a crea un tabel pornind de la rezultatul unei instrucțiuni SELECT au fost introduse în versiunea MySQL 3.23.

O opțiune *creare_definitie* poate fi o declarație de coloană, o definiție de index, o clauză FOREIGN KEY, o opțiune *referinta_definitie* sau o clauză CHECK. Ultimele trei sunt analizate pentru a se determina compatibilitatea cu alte sisteme de baze de date, dar de regulă sunt ignorate.

O declarație de coloană *declaratie_coloana* începe cu un nume de coloană *nume_coloana* și de un tip *tip_coloana*, după care pot urma numeroase cuvinte cheie opționale. Tipul de coloană poate fi oricare dintre tipurile prezentate în Anexa B, „Referință de tipuri de coloane". Pentru a determina atributele specifice unui tip anume, atribute aplicabile coloanelor pe care doriți să le declarați, examinați anexa respectivă. Cuvintele cheie opționale care pot apărea după numele coloanei sunt următoarele:

- NULL sau NOT NULL

  Arată că respectiva coloană poate conține sau nu valori NULL. Dacă nici una din aceste opțiuni nu este specificată, NULL este opțiunea prestabilită.

- DEFAULT valoare_prestabilita

  Indică valoarea prestabilită a coloanei. Nu se poate folosi pentru tipuri BLOB sau TEXT. Dacă nu este specificată nici o valoare prestabilită, atunci va fi atribuită o asemenea valoare. Pentru coloane care pot accepta valori NULL, valoarea prestabilită este NULL. Pentru coloanele care nu pot accepta valori NULL, valoarea prestabilită este atribuită după cum urmează:

  - Pentru coloanele numerice, valoarea prestabilită este 0, cu excepția coloanelor AUTO_INCREMENT. Pentru AUTO_INCREMENT, valoarea prestabilită este următorul număr din secvența de numere aferentă coloanei.
  - Pentru tipuri dată-oră altele decât TIMESTAMP, valoarea prestabilită este valoarea „zero" aferentă tipului respectiv (de exemplu "0000-00-00" pentru tipul DATE). Pentru TIMESTAMP, valoarea prestabilită este data și ora curentă.
  - Pentru tipuri șir, altele decât ENUM, valoarea prestabilită este șirul vid. Pentru ENUM, valoarea prestabilită este primul element din enumerare.

- AUTO_INCREMENT

  Acest cuvânt cheie se aplică numai tipurilor de coloane cu valori întregi. O coloană AUTO_INCREMENT este mai deosebită, în sensul că atunci când introduceți NULL într-o asemenea coloană, valoarea inserată efectiv este cu o unitate mai mare decât valoarea curentă maximă din coloană. Valorile AUTO_INCREMENT pornesc în mod prestabilit de la 1; prima valoare poate fi specificată în mod explicit cu opțiunea de tabel AUTO_INCREMENT. De asemenea, coloanele trebuie specificate sub formă de index UNIQUE sau PRIMARY KEY și trebuie să fie de tipul NOT NULL. Într-un tabel poate exista cel mult o coloană AUTO_INCREMENT.

- PRIMARY KEY

  Arată că respectiva coloană este de tip PRIMARY KEY.

- UNIQUE

  Arată că respectiva coloană este un index UNIQUE. Acest atribut poate fi specificat începând de la versiunea MySQL 3.23.

Clauzele PRIMARY KEY, UNIQUE, INDEX și KEY specifică indexuri; PRIMARY KEY și UNIQUE specifică indexuri care trebuie să conțină valori unice. INDEX și KEY sunt sinonime și specifică indexuri care pot conține valori duplicate. Indexul este bazat pe coloanele denumite în *coloane_index*, care trebuie să fie coloane în tabelul *nume_tabel*. Dacă sunt precizate mai multe coloane, acestea trebuie separate prin virgule. Pentru coloanele CHAR și VARCHAR, puteți indexa un prefix al coloanei, folosind sintaxa *nume_coloana(n)* pentru a indexa primele n caractere ale valorilor din coloane. Pentru coloanele BLOB și TEXT, *trebuie* să specificați o valoare a prefixului; nu puteți indexa întreaga coloană. Dacă numele indexului *nume_index* nu este specificat, atunci se va selecta un nume în mod automat, în funcție de numele primei coloane indexate.

Anterior versiunii MySQL 3.23, coloanele indexate trebuie declarate de tip NOT NULL. Începând de la versiunea MySQL 3.23, numai coloanele din indexurile PRIMARY KEY trebuie declarate NOT NULL.

Clauza *optiuni_tabel* este disponibilă începând de la versiunea MySQL 3.23. În prezent, poate conține una sau mai multe dintre opțiunile din lista următoare. Dacă există mai multe opțiuni, acestea nu trebuie separate prin virgule. Fiecare specificator se aplică tuturor tipurilor de tabele, cu excepția situațiilor când există alte precizări.

- AUTO_INCREMENT = *n*

  Prima valoare AUTO_INCREMENT care va fi generată pentru tabel (valabil numai pentru tabelele MyISAM).

- AVG_ROW_LENGTH = *n*

  Lungimea medie aproximativă a rândurilor tabelului dumneavoastră. Această opțiune este folosită de către MySQL pentru a determina dimensiunea maximă a fișierului de date și este necesară numai pentru tabele care conțin coloane BLOB sau TEXT, când tabelele respective pot deveni mai mari de 4GB.

- CHECKSUM = {0 | 1}

  Dacă această opțiune are valoarea 1, MySQL folosește o sumă de control pentru fiecare rând de tabel. Există un mic dezavantaj pentru actualizările aduse tabelului, dar astfel se îmbunătățește procesul de verificare a tabelului (valabil numai pentru tabelele MyISAM).

- COMMENT = "*sir*"

  Un comentariu pentru tabel. Lungimea maximă este de 60 de caractere.

- DELAY_KEY_WRITE = {0 | 1}

  Dacă această opțiune are valoarea 1, memoria cache a indexului este golită numai ocazional pentru tabel, nu după fiecare operație de inserție (valabil numai pentru tabelele MyISAM).

- MAX_ROWS = *n*

  Numărul maxim de rânduri pe care intenționați să-l stocați în tabel.

- MIN_ROWS = *n*

  Numărul minim de rânduri pe care intenționați să-l stocați în tabel.

- PACK_KEYS = {0 | 1}

  Dacă această opțiune este configurată la valoarea 1, blocurile de index sunt împachetate la un procentaj mai mare decât cel normal. Efectul normal este o penalizare la actualizare și o îmbunătățire a performanțelor operațiilor de regăsire. Dacă opțiunea are valoarea 0, blocurile de index nu sunt împachetate în mod special. În acest caz, sunt împachetate numai indexurile cu valori cheie de tip CHAR sau VARCHAR cu o lungime de minimum 8 caractere (valabil numai pentru tabelele MyISAM și ISAM).

- PASSWORD = "*sir*"

  Specifică o parolă pentru criptarea fișierului de descriere a tabelului. În mod normal, această operație nu are nici un efect; opțiunea este activată numai pentru anumiți clienți care beneficiază de un contract de asistență.

- TYPE = {ISAM | MYISAM | HEAP}

  Specifică formatul de stocare în tabel. Caracteristicile acestor formate de stocare sunt descrise în secțiunea „Specificatori ai tipului de stocare în tabel" din capitolul 3, „Sintaxa și utilizarea SQL în MySQL". Formatul prestabilit pentru MySQL începând de la versiunea 3.23 este MyISAM. Anterior versiunii MySQL 3.23, instrucțiunea CREATE TABLE crea întotdeauna tabele în formatul ISAM.

Dacă este specificată o clauză *instructiune_select* (sub forma unei interogări SELECT), tabelul este creat folosind conținutul setului de rezultate returnat de interogare. Rândurile care reproduc valorile dintr-un index unic sunt fie ignorate, fie înlocuiesc rândurile existente, în funcție de specificarea uneia dintre opțiunile IGNORE sau REPLACE. Dacă nu este specificată nici una dintre aceste opțiuni, se va produce o eroare, iar toate înregistrările rămase sunt ignorate.

Să creăm un tabel cu trei coloane. Coloana id este de tip PRIMARY KEY, iar cele două coloane de nume sunt indexate împreună:

```
CREATE TABLE client
(
   id SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,
   nume CHAR(30) NOT NULL,
   prenume CHAR(20) NOT NULL,
   PRIMARY KEY (id),
   INDEX (nume, prenume)
)
```

Creați un tabel temporar și transformați-l în tabel HEAP (stocat în memorie), pentru un acces mai rapid:

```
CREATE TEMPORARY TABLE tmp_tabel
   (id MEDIUMINT NOT NULL UNIQUE, nume CHAR(40))
   TYPE = HEAP
```

Creați un tabel ca o copie a unui alt tabel:

```
CREATE TABLE prez_copie SELECT * FROM presedinte
```

Creați un tabel folosind numai o parte a unui alt tabel:

```
CREATE TABLE prez_traiesc SELECT nume, prenume, nastere
   FROM presedinte WHERE deces IS NULL
```

Dacă declarațiile de creare sunt specificate pentru un tabel creat și populat prin intermediul unei instrucțiuni SELECT finale, declarațiile sunt aplicate după ce conținutul tabelului a fost inserat în tabel. De exemplu, puteți declara că o coloană selectată trebuie transformată în PRIMARY KEY:

```
CREATE TABLE tabel_nou (PRIMARY KEY(a)) SELECT a, b, c FROM tabel_vechi
```

DELETE

```
DELETE [LOW_PRIORITY] FROM nume_tabel
   [WHERE unde_expr] [LIMIT n]
```

Șterge rânduri din tabelul *nume_tabel*. Rândurile șterse sunt cele care corespund condițiilor specificate în expresia *unde_expr*:

```
DELETE FROM puncte WHERE eveniment_id = 14
DELETE FROM membru WHERE expirare < CURRENT_DATE
```

În cazul în care clauza WHERE este omisă, *toate înregistrările din tabel vor fi șterse.*

Instrucțiunea DELETE returnează numărul de înregistrări șterse, cu excepția situației când ștergeți toate înregistrările prin nespecificarea clauzei WHERE. MySQL optimizează acest caz special prin re-crearea fișierelor de date și a fișierelor index pornind de la zero, nu prin ștergerea rând cu rând a înregistrărilor. Această operație este extrem de rapidă, dar nu pune la dispoziție numărul rândurilor afectate și returnează valoarea zero. Pentru a obține un număr autentic, specificați o clauză WHERE care returnează toate înregistrările. De exemplu:

```
DELETE FROM nume_tabel WHERE 1 > 0
```

Totuși, apare o diminuare sensibilă a performanțelor.

Dacă este specificată opțiunea LOW_PRIORITY, execuția instrucțiunii este amânată până când nici un client nu mai citește din tabel. Opțiunea LOW_PRIORITY a fost introdusă în MySQL versiunea 3.22.5.

Dacă este dată clauza LIMIT, valoarea n specifică numărul maxim de rânduri care vor fi șterse. Clauza LIMIT a fost introdusă în MySQL versiunea 3.22.7.

DESCRIBE

```
{DESCRIBE | DESC} nume_tabel {nume_coloana | model}
```

Această instrucțiune furnizează aceleași informații ca și SHOW COLUMNS. Pentru mai multe informații, consultați descrierea instrucțiunii SHOW.

DROP DATABASE

```
DROP DATABASE [IF EXISTS] nume_baza_de_date
```

Elimină (șterge) baza de date specificată. După ce ați șters o bază de date, aceasta a dispărut pe vecie, deci fiți atent. Instrucțiunea eșuează dacă baza de date nu există (cu excepția cazului când specificați IF EXISTS) sau dacă nu aveți privilegiul adecvat. Opțiunea IF EXISTS a fost introdusă în versiunea MySQL 3.22.2.

O bază de date este reprezentată într-un catalog amplasat în catalogul de date. Dacă în catalogul respectiv ați inserat fișiere care nu reprezintă tabele, fișierele respective nu vor fi șterse de instrucțiunea DROP DATABASE, caz în care nici catalogul bazei de date nu va fi eliminat.

DROP FUNCTION

```
DROP FUNCTION nume_functie
```

Elimină o funcție definită de utilizator care a fost încărcată anterior cu instrucțiunea CREATE FUNCTION.

DROP INDEX

```
DROP INDEX nume_index ON nume_tabel
```

Șterge indexul *nume_index* din tabelul *nume_tabel*. Această instrucțiune este manipulată asemenea unei instrucțiuni ALTER TABLE DROP INDEX. Pentru detalii, examinați descrierea instrucțiunii ALTER TABLE. Instrucțiunea DROP INDEX a fost introdusă în MySQL versiunea 3.22.

DROP TABLE

```
DROP TABLE [IF EXISTS] nume_tabel [, nume_tabel] ...
```

Șterge tabelul sau tabelele specificate din baza de date căreia îi aparțin acestea. Dacă este dată clauza IF EXISTS, ștergerea unui tabel inexistent nu este o eroare. Clauza IF EXISTS a fost introdusă în MySQL versiunea 3.22.2.

EXPLAIN

```
EXPLAIN nume_tabel
EXPLAIN instructiune_select
```

Prima formă a acestei instrucțiuni este echivalentă cu SHOW COLUMNS FROM *nume_tabel*. Pentru mai multe informații, examinați descrierea instrucțiunii SHOW.

Cea de-a doua formă a instrucțiunii EXPLAIN furnizează informații referitoare la modul în care MySQL va executa instrucțiunea SELECT care urmează după cuvântul cheie EXPLAIN.

```
EXPLAIN SELECT puncte.* FROM puncte, eveniment
  WHERE puncte.eveniment_id = eveniment.eveniment_id
  AND eveniment.eveniment_id = 14
```

Datele de ieșire ale instrucțiunii EXPLAIN sunt alcătuite din unul sau mai multe rânduri, care conțin următoarele coloane:

- table

  Tabelul la care se referă rândul de ieșire.

- type

  Tipul de unire pe care îl va efectua MySQL. Tipurile posibile sunt următoarele, în ordine descrescătoare a calității: system, const, eq_ref, ref, range, index și ALL. Tipurile mai bune sunt mai restrictive, ceea ce înseamnă că MySQL trebuie să examineze mai puține rânduri din tabel atunci când execută operația de regăsire.

- possible_keys

  Indexurile pe care MySQL le consideră adecvate pentru găsirea rândurilor din tabelul denumit în coloana row. Valoarea NULL arată că nu au fost găsite indexuri.

- key

  Indexul pe care îl va folosi efectiv MySQL pentru găsirea rândurilor din tabel. Valoarea NULL arată că nu se va folosi nici un index.

- key_len

  Porțiunea din index care va fi folosită. Aceasta poate fi mai mică decât lungimea totală a rândului de index, dacă MySQL va folosi un prefix de stânga al indexului.

- `ref`

  Valorile cu care MySQL va compara valorile din index. Cuvântul `const` sau expresia `???` arată că termenul de comparație este o constantă; un nume de coloană indică o comparație între coloane.

- `rows`

  O valoare estimativă a numărului de rânduri din tabel pe care MySQL trebuie să le examineze pentru a executa interogarea. Produsul valorilor din această coloană este o valoare aproximativă a numărului total de combinații de rânduri care trebuie examinate din toate tabelele.

- `Extra`

  Opțiunea `Only index` arată că MySQL poate regăsi informațiile aferente tabelului folosind numai informațiile din index, fără a examina fișierul de date. Opțiunea `where used` indică utilizarea informațiilor din clauza `WHERE` a instrucțiunii `SELECT`.

FLUSH

```
FLUSH optiune_golire [, optiune_golire] ...
```

Golește diferitele zone cache interne folosite de server. Opțiunea `optiune_golire` poate fi oricare dintre articolele din lista următoare:

- `HOSTS`

  Golește zona cache aferentă gazdelor.

- `LOGS`

  Golește fișierele jurnal, închizându-le și redeschizându-le.

- `PRIVILEGES`

  Reîncarcă tabelele de acordare.

- `STATUS`

  Reinițializează variabilele de stare.

- `TABLES`

  Închide toate tabelele deschise în zona cache a tabelului.

Instrucțiunea `FLUSH` a fost introdusă în MySQL 3.22.9. Opțiunea de golire `STATUS` a fost introdusă în MySQL versiunea 3.22.11.

GRANT

```
GRANT tip_priv [(lista_coloane)][, tip_priv [(lista_coloane)] ] ...
   ON {*.* | * | nume_db.* | nume_db.nume_tabel | nume_tabel}
   TO utilizator [, utilizator ] ...
   [WITH GRANT OPTION]
```

Instrucțiunea `GRANT` acordă privilegii de acces unuia sau mai multor utilizatori MySQL. Valoarea `tip_priv` specifică privilegiile care vor trebui acordate și constă din una sau mai multe tipuri de privilegii separate prin virgulă, selectate din următoarea listă:

Specificator de privilegiu	Operație permisă de privilegiu
ALTER	Modifică tabele și indexuri
CREATE	Creează baze de date și tabele
DELETE	Șterge din tabele înregistrări existente
DROP	Șterge (elimină) baze de date și tabele
INDEX	Creează sau șterge indexuri
INSERT	Inserează înregistrări noi în tabele
REFERENCES	Nefolosit
SELECT	Regăsește înregistrări existente din tabele
UPDATE	Modifică înregistrări existente din tabele
FILE	Citește și scrie fișiere pe server
PROCESS	Vizualizează informații despre firele care se execută în cadrul serverului sau suprimă firele
RELOAD	Reîncarcă tabelele de acordare sau golește jurnalele, zona cache aferentă gazdelor sau zona cache a tabelelor
SHUTDOWN	Oprește serverul
ALL	Orice acțiune; sinonim cu `ALL PRIVILEGES`
USAGE	Un privilegiu special „fără privilegii"

Vă puteți întotdeauna vizualiza sau suprima propriile fire. Privilegiul `PROCESS` vă permite să vizualizați sau să suprimați orice fir.

Clauza `ON` specifică domeniul de acordare a privilegiilor, așa cum se poate vedea în tabelul următor:

Specificator de privilegii	Nivelul de aplicare a privilegiilor
`*.*`	Privilegii globale; toate bazele de date, toate tabelele
`*`	Privilegii globale, dacă nu a fost selectată nici o bază de date prestabilită; privilegii la nivel de bază de date pentru baza de date curentă, în caz contrar
`nume_db.*`	Privilegii la nivel de bază de date; toate tabelele din baza de date specificată
`nume_db.nume_tabel`	Privilegii la nivel de tabel; toate coloanele din tabelul specificat
`nume_tabel`	Privilegii la nivel de tabel; toate coloanele din tabelul denumit în baza de date prestabilită

Când este specificat un tabel în clauza `ON`, privilegiile trebuie să fie la nivel de coloană; acest lucru se obține prin denumirea uneia sau mai multor coloane separate prin virgulă în clauza `lista_coloane`.

Clauza `TO` specifică unul sau mai mulți utilizatori cărora trebuie să li se acorde privilegiile. Fiecare `utilizator` constă dintr-un specificator în formatul *`nume_utilizator@nume_gazda`*, cu o clauză opțională `IDENTIFIED BY`. *`nume_utilizator`* poate fi un nume sau un șir vid (`""`); ultimul specifică un utilizator anonim. *`nume_gazda`* poate fi specificat ca `localhost`, ca un

nume de gazdă, ca o adresă IP sau ca un model care corespunde unui nume de domeniu sau unui număr de rețea. Caracterele dintr-un model sunt % și _, cu aceeași semnificație ca pentru operatorul LIKE. Un *nume_utilizator* specificat singur, fără nici un nume de gazdă, este echivalent cu notația *nume_utilizator*@%. Începând de la versiunea MySQL 3.23, nume_utilizator poate fi o pereche număr IP-mască de rețea scrise sub forma *n.n.n.n/m*, unde *n.n.n.n* indică adresa IP, iar *m* indică numărul de biți ce trebuie utilizați pentru numărul rețelei.

Clauza IDENTIFIED BY, dacă este dată, atribuie utilizatorului o parolă. Parola trebuie specificată în format text simplu, fără a folosi funcția PASSWORD, spre deosebire de modul în care sunt specificate parolele pentru instrucțiunea SET PASSWORD. Dacă utilizatorul există deja și opțiunea IDENTIFIED BY este specificată, noua parolă o va înlocui pe cea veche. Parola existentă rămâne, în caz contrar, nemodificată.

Instrucțiunea GRANT a fost introdusă în versiunea MySQL 3.22.11.

Instrucțiunile următoare exemplifică unele moduri de utilizare a instrucțiunii GRANT. Pentru alte exemple, consultați capitolul 11, „Administrarea generală a sistemului MySQL".

Creați un utilizator paul cu acces la toate tabelele din baza de date samp_db de la orice gazdă. Următoarele două instrucțiuni sunt echivalente, deoarece omiterea numelui gazdei în identificatorul utilizator este echivalentă cu specificarea caracterului %:

```
GRANT ALL ON samp_db.* TO paul IDENTIFIED BY "secret"
GRANT ALL ON samp_db.* TO paul@% IDENTIFIED BY "secret"
```

Creați un utilizator care dispune numai de privilegii de citire pentru tabelele din baza de date menagerie. Utilizatorul se poate conecta de la orice gazdă din domeniul xyz.com:

```
GRANT SELECT ON menagerie.* TO numai_ochi@%.xyz.com
   IDENTIFIED BY "rugina"
```

Creați un utilizator cu privilegii complete, dar numai pentru tabelul membru din baza de date samp_db. Utilizatorul se poate conecta de la o singură gazdă:

```
GRANT ALL ON samp_db.membru TO membru_mgr@boa.snake.net
   IDENTIFIED BY "gogoasa"
```

Creați un super-utilizator care poate face orice, inclusiv să acorde privilegii altor utilizatori, dar care trebuie să se conecteze de la gazda locală:

```
GRANT ALL ON *.* TO maresitare@localhost IDENTIFIED BY "homer"
   WITH GRANT OPTION
```

Creați un utilizator anonim al bazei de date menagerie, care se poate conecta de la gazda locală, fără parolă:

```
GRANT ALL ON menagerie.* TO ""@localhost
```

INSERT

```
INSERT [LOW PRIORITY | DELAYED] [IGNORE] [INTO]
   nume_tabel [(lista_coloane)] VALUES (expr [, expr] ...) [, (...)] ...
INSERT [LOW PRIORITY | DELAYED] [IGNORE] [INTO]
   nume_tabel [(lista_coloane)] SELECT ...
INSERT [LOW PRIORITY | DELAYED] [IGNORE] [INTO]
   nume_tabel SET nume_coloana=expr [,nume_coloana=expr] ...
```


Inserează rânduri într-un tabel nume_tabel existent și returnează numărul de rânduri inserate. Cuvântul cheie INTO este obligatoriu în versiunile anterioare versiunii MySQL 3.22.5.

LOW_PRIORITY determină amânarea execuției instrucțiunii până când nici un client nu mai citește din tabel. Opțiunea LOW_PRIORITY a fost introdusă în MySQL versiunea 3.22.5.

Opțiunea DELAYED determină plasarea rândurilor într-o coadă în vederea unei inserții ulterioare, iar instrucțiunea returnează imediat, astfel încât clientul își poate continua execuția, fără să mai aștepte. Totuși, în acest caz, funcția LAST_INSERT_ID() nu va returna valoarea AUTO_INCREMENT aferentă pentru orice coloană AUTO_INCREMENT din tabel. Inserțiile cu ajutorul opțiunii DELAYED au fost introduse în MySQL versiunea 3.22.15.

Dacă este specificată opțiunea IGNORE, rândurile care reproduc valorile cheilor unice din rândurile existente sunt ignorate. Dacă valorile duplicate apar fără IGNORE, se va produce o eroare și nu se mai inserează nici un rând. Opțiunea IGNORE a fost introdusă în MySQL 3.22.10.

Prima formă a funcției INSERT necesită o listă VALUES() care specifică toate valorile care vor fi inserate. Dacă nu este dată nici o listă de coloane, lista VALUES() trebuie să specifice o valoare pentru fiecare coloană din tabel. Dacă este dată o listă de coloane, alcătuită din unul sau mai multe nume de coloană, separate prin virgulă, în lista VALUES() trebuie specificată câte o valoare pentru fiecare coloană. Coloanele care nu sunt specificate în lista de coloane își primesc valorile lor prestabilite. Începând de la versiunea MySQL 3.22.5, se pot specifica mai multe liste cu valori, permițând inserția mai multor rânduri folosind o singură instrucțiune INSERT. Începând cu versiunea MySQL 3.23.3, lista VALUES() poate fi vidă, ceea ce are ca rezultat inserția unui rând pentru care fiecare coloană își primește valoarea prestabilită.

A doua formă a instrucțiunii INSERT regăsește înregistrările în conformitate cu instrucțiunea SELECT și le inserează în nume_tabel. Instrucțiunea SELECT trebuie să selecteze toate coloanele din nume_tabel, respectiv toate coloanele din lista_coloane, dacă se specifică o listă de coloane. Când este specificată o listă de coloane, toate coloanele care nu sunt specificate în listă își vor primi valorile lor prestabilite.

Cea de-a treia formă a instrucțiunii INSERT, disponibilă începând cu versiunea MySQL 3.22.10, inserează coloane denumite în clauza SET în valorile date de expresiile corespunzătoare. Coloanele care nu sunt denumite își primesc valorile prestabilite.

```
INSERT INTO absente (elev_id, data) VALUES(14,"1999-11-03"),(34,NOW())
INSERT INTO absente SET elev_id = 14, data = "1999-11-03"
INSERT INTO absente SET elev_id = 34, data = NOW()
INSERT INTO puncte (elev_id, puncte, eveniment_id)
   SELECT elev_id, 100 AS puncte, 15 AS eveniment_id FROM elev
```

KILL

```
KILL fir_id
```

Suprimă firul de server cu identificatorul de fir *fir_id*. Trebuie să aveți privilegiul PROCESS pentru a suprima firul, cu excepția situației când firul respectiv vă aparține.

Comanda mysqladmin kill execută aceeași operație, dar permite specificarea în linia de comandă a mai multor valori de identificatori de fir. Instrucțiunea KILL permite numai un singur identificator.

Această instrucțiune a fost introdusă în versiunea MySQL 3.22.9.

LOAD DATA

```
LOAD DATA [LOW PRIORITY] [LOCAL] INFILE 'nume_fisier'
   [IGNORE | REPLACE]
   INTO TABLE nume_tabel
   optiuni_import
   [IGNORE n LINES]
   [(lista_coloane)]
```

Instrucțiunea LOAD DATA citește înregistrări din fișierul *nume_fisier* și le încarcă în cantități mari în tabelul *nume_tabel*. Operația este mai rapidă decât INSERT.

LOW PRIORITY determină amânarea execuției instrucțiunii până când nici un client nu mai citește din tabel. Această opțiune a fost introdusă în versiunea MySQL 3.23.

În mod normal, fișierul este citit direct de serverul situat pe gazda serverului. În acest caz, trebuie să aveți privilegiul FILE și fișierul trebuie să se găsească fie în catalogul bazei de date prestabilite, fie trebuie să fie universal accesibil la citire. Dacă se specifică opțiunea LOCAL, clientul citește fișierul pe gazda clientului și trimite conținutul fișierului la server, prin rețea. În acest caz, privilegiul FILE nu este obligatoriu. Opțiunea LOCAL a devenit funcțională în versiunea MySQL 3.22.15.

Când opțiunea LOCAL nu este specificată, serverul găsește fișierul după cum urmează:

- Dacă *nume_fisier* este o cale de acces absolută, serverul caută fișierul începând de la catalogul rădăcină.
- Dacă *nume_fisier* este o cale de acces relativă, interpretarea depinde de faptul dacă numele conține sau nu o singură componentă. În caz afirmativ, serverul caută fișierul în catalogul bazei de date prestabilite. Dacă numele fișierului conține mai multe componente, serverul caută fișierul începând de la catalogul de date al serverului.

  Dacă opțiunea LOCAL este dată, numele fișierului este interpretat astfel:

  - Dacă *nume_fisier* este o cale de acces absolută, clientul caută fișierul începând de la catalogul rădăcină.
  - Dacă *nume_fisier* este o cale de acces relativă, clientul caută fișierul începând de la catalogul dumneavoastră curent.

Rândurile care conțin valori duplicate ale unui index unic sunt fie ignorate, fie înlocuiesc rândurile existente, în funcție de specificarea opțiunii IGNORE sau REPLACE. Dacă nici una dintre aceste opțiuni nu este specificată, se va produce o eroare și toate înregistrările rămase vor fi ignorate.

Clauza *optiuni_import* indică formatul datelor. Opțiunile disponibile în această clauză se aplică și clauzei *optiuni_export* aferente instrucțiunii SELECT ... INTO OUTFILE. Sintaxa pentru *optiuni_import* este următoarea:

```
[FIELDS
   [TERMINATED BY 'sir']
   [[OPTIONALLY] ENCLOSED BY 'caracter']
   [ESCAPED BY 'caracter']]
[LINES TERMINATED BY 'sir']
```


Valorile *sir* și *caracter* pot include următoarele secvențe escape pentru indicarea caracterelor speciale:

Secvență	Semnificație
\0	ASCII 0
\b	Ștergere caracter anterior
\n	Caracter linie nouă
\r	Retur de car
\s	Spațiu
\t	Caracter de tabulare
\'	Ghilimele simple
\"	Ghilimele duble
\\	Backslash

Începând de la versiunea MySQL 3.22.10, mai puteți folosi și constante hexazecimale pentru a indica anumite caractere arbitrare. De exemplu, LINES TERMINATED BY 0x02 arată că liniile se încheie prin caractere Ctrl-B (ASCII 2).

Dacă este precizată opțiunea FIELDS, atunci trebuie dată cel puțin una dintre clauzele TERMINATED BY, ENCLOSED BY sau ESCAPED BY, iar clauzele date trebuie specificate în ordinea indicată. Dacă opțiunile FIELDS și LINES sunt ambele indicate, opțiunea FIELDS trebuie să preceadă opțiunea LINES. Componentele clauzei FIELDS sunt folosite după cum urmează:

- TERMINATED BY arată caracterul sau caracterele care delimitează valorile din cadrul unei linii.
- ENCLOSED BY precizează un caracter de citare care este eliminat de la extremitățile valorilor câmpurilor, dacă este găsit. Această operație are loc indiferent dacă clauza OPTIONALLY este prezentă sau nu. Pentru ieșire (SELECT ... INTO OUTFILE), caracterul ENCLOSED BY este folosit pentru a încadra valorile câmpurilor în liniile de ieșire. Dacă este dată clauza OPTIONALLY, valorile sunt încadrate între ghilimele numai pentru coloanele CHAR și VARCHAR.

  Pentru a include o instanță a caracterului ENCLOSED BY într-o valoare a unui câmp de intrare, aceasta trebuie fie dublată, fie precedată de caracterul ESCAPED BY. În caz contrar, va fi interpretat ca indicând sfârșitul câmpului. Pentru datele de ieșire, instanțele caracterului ENCLOSED BY din interiorul valorilor câmpurilor sunt precedate de caracterul ESCAPED BY.
- Caracterul ESCAPED BY este folosit pentru a modifica semnificația caracterelor speciale (escaping). În exemplele următoare, vom presupune că acest caracter escape este caracterul backslash (\). Pentru intrare, combinația \N (backslash-N) este interpretată ca NULL, iar \0 (backslash-ASCII 0) este interpretată ca un octet cu valoare zero. Pentru alte caractere, caracterul escape este eliminat și caracterul următor este folosit literal. De exemplu, \" este interpretat drept ghilimele duble, chiar dacă valorile câmpurilor sunt încadrate între ghilimele simple.

Pentru ieșire, caracterul escape este folosit pentru codificarea octeților de valori NULL și zero sub forma \N, respectiv \0. De asemenea, instanțele caracterelor ESCAPED BY și ENCLOSED BY sunt precedate de caracterul escape, așa cum sunt și primele caractere ale șirurilor de terminare ale câmpurilor și liniilor. În cazul în care caracterul ESCAPED BY este vid (ESCAPED BY ''), nu se va produce nici o modificare a semnificației caracterelor speciale. Pentru a specifica \ drept caracter escape, dublați-l (ESCAPED BY \\).

Valoarea LINES TERMINATED BY specifică un caracter sau caractere care semnifică sfârșitul liniilor.

Dacă nu este dată nici una din clauzele FIELDS sau LINES, valorile prestabilite se prezintă ca și cum le-ați fi specificat astfel:

```
FIELDS
   TERMINATED BY '\t'
   ENCLOSED BY ''
   ESCAPED BY '\\'
LINES TERMINATED BY '\n'
```

Cu alte cuvinte, câmpurile din cadrul unei linii sunt delimitate prin tabulatori, fără a fi încadrate între ghilimele, backslash-ul este tratat drept caracter escape, iar liniile sunt terminate prin caractere linie nouă.

Dacă valorile TERMINATED BY și ENCLOSED BY pentru clauza FIELDS sunt ambele vide, se va folosi un format cu rânduri fixe, fără delimitatori între câmpuri. Valorile câmpurilor sunt citite (respectiv scrise, pentru ieșire) folosind lățimile de afișare ale coloanelor. De exemplu, coloanele VARCHAR(15) și MEDIUMINT(5) sunt citite sub formă de câmpuri de 15, respectiv 5 caractere pentru intrare. Pentru ieșire, coloanele sunt scrise folosind 15 caractere, respectiv 5 caractere. Valorile NULL sunt scrise sub formă de șiruri de spații.

Dacă este dată clauza IGNORE n LINES, primele n linii ale datelor de intrare sunt ignorate. De exemplu, dacă fișierul dumneavoastră de date are un rând cu antete de coloană pe care nu doriți să-l inserați în tabelul bazei de date, puteți folosi comanda IGNORE 1 LINES:

```
LOAD DATA LOCAL INFILE "tabelul_meu.txt"
   INTO TABLE tabelul_meu IGNORE 1 LINES
```

Dacă nu este specificată nici o listă de coloane, se va presupune că liniile de intrare conțin câte o valoare pentru fiecare coloană din tabel. Dacă o listă constă din una sau mai multe nume de coloane separate prin virgulă, liniile de intrare trebuie să conțină o valoare pentru fiecare coloană denumită. Coloanele care nu sunt specificate în listă sunt configurate la valorile lor prestabilite. Dacă dintr-o linie de intrare lipsește numărul de valori specificat, coloanele din care valorile lipsesc își vor primi valorile prestabilite.

Dacă aveți un fișier text delimitat prin tabulatori pe care l-ați creat în Windows, puteți folosi separatorul de coloane prestabilit, dar probabil că liniile se termină prin perechi retur de car-linie nouă. Pentru a încărca fișierul, specificați un alt caracter de terminare a liniei (\r indică un retur de car, iar \n indică un caracter linie nouă):

```
LOAD DATA LOCAL INFILE "tabelul_meu.txt"
   INTO TABLE tabelul_meu LINES TERMINATED BY "\r\n"
```


Din păcate, pentru fișierele create în Windows, vă puteți alege cu o înregistrare eronat în baza de date dacă programul care a creat fișierul de date folosește ciudata convenți din MS-DOS de inserție a caracterului Ctrl-Z la sfârșitul unui fișier, pentru a indic sfârșitul fișierului. Fie scrieți fișierul folosind un program care nu procedează astfel, fi ștergeți înregistrarea după încărcarea fișierului.

Fișierele cu valori separate prin virgulă (CSV) au virgule între câmpuri, iar câmpurile po fi încadrate între ghilimele duble. Presupunând că liniile se încheie prin caractere lini nouă, instrucțiunea LOAD DATA pentru încărcarea unui atare fișier se prezintă astfel:

```
LOAD DATA LOCAL INFILE "tabelul_meu.txt"
   INTO TABLE tabelul_meu LINES TERMINATED BY "," ENCLOSED BY "\""
```

Instrucțiunea următoare prezintă modul de citire a unui fișier pentru care câmpuril sunt separate prin caractere Ctrl-A (ASCII 1), iar liniile se încheie cu caractere Ctrl-I (ASCII 2):

```
LOAD DATA LOCAL INFILE "tabelul_meu.txt" INTO TABLE tabelul_meu
   FIELDS TERMINATED BY 0x01 LINES TERMINATED BY 0x02
```

LOCK TABLES

```
LOCK TABLES lista_blocare
```

Obține o blocare asupra tabelelor menționate în *lista_blocare*, așteptând, dacă est necesar, până la acumularea tuturor blocărilor. LOCK TABLE este sinonim cu LOCK TABLES

lista_blocare conține unul sau mai multe articole separate prin virgulă, unde fiecar articol respectă următoarea sintaxă:

```
nume_tabel [AS nume_alias] {READ | [LOW_PRIORITY] WRITE}
```

READ indică o blocare pentru citire. Aceasta îi blochează pe alți clienți care doresc să scrie în tabel, dar le permite să citească tabelul. WRITE indică o blocare la scriere. Aceasta îi blochează pe toți ceilalți clienți, indiferent dacă aceștia doresc să citească sau să scrie în tabel. O cerere de blocare LOW_PRIORITY WRITE permite altor programe de citire să citească tabelul, în cazul în care cererea așteaptă un alt client, care deja citește tabelul. Blocarea este obținută doar atunci când nu mai există programe care citesc tabelul.

LOCK TABLES permite specificarea unui alias, astfel încât puteți bloca un tabel la care veți face referire folosind aliasul respectiv într-o interogare ulterioară. Dacă folosiți de mai multe ori un tabel într-o interogare, trebuie să obțineți o blocare pentru fiecare instanță a tabelului, blocând aliasurile după necesități.

LOCK TABLES anulează toate blocările existente pe care le deținеți la un moment dat. Astfel, pentru a bloca mai multe tabele, trebuie să le blocați pe toate folosind o singură instrucțiune LOCK TABLES. Toate blocările deținute de un client atunci când acesta își termină execuția vor fi anulate automat.

Blocările de tip LOW_PRIORITY au fost introduse în versiunea MySQL 3.22.5.

```
LOCK TABLES elev READ, puncte WRITE, eveniment READ
LOCK TABLE membru READ
```

OPTIMIZE TABLE

```
OPTIMIZE TABLE nume_tabel
```

Recuperează spațiul irosit într-un tabel. Instrucțiunile DELETE, REPLACE și UPDATE pot avea ca rezultat apariția unor regiuni de spațiu nefolosit într-un tabel, mai ales în tabelele cu rânduri de lungime variabilă.

Instrucțiunea OPTIMIZE TABLE elimină spațiul irosit, reducând dimensiunea tabelului. Această instrucțiune a fost introdusă în MySQL 3.22.7. Alte opțiuni de întreținere a tabelului sunt programate a fi introduse în seria de versiuni MySQL 3.23, deci examinați manualul de referință MySQL pe suport electronic pentru a descoperi ultimele adăugiri.

REPLACE

```
REPLACE [LOW_PRIORITY | DELAYED] [INTO]
   nume_tabel [(lista_coloane)] VALUES (expr [, expr] ...) [, (...)] ...
REPLACE [LOW_PRIORITY | DELAYED] [INTO]
   nume_tabel [(lista_coloane)] SELECT ...
REPLACE [LOW_PRIORITY | DELAYED] [INTO]
   nume_tabel SET nume_coloana=expr [,nume_coloana=expr] ...
```

Instrucțiunea REPLACE este similară cu INSERT, cu deosebirea că, dacă un rând care trebuie inserat are o valoare a unui index UNIQUE care reproduce valoarea dintr-un rând deja existent în tabel, vechea valoare este ștearsă anterior introducerii noii valori. Din acest motiv, nu există nici o clauză IGNORE în sintaxa instrucțiunii REPLACE.

Pentru mai multe informații, vezi prezentarea instrucțiunii INSERT.

REVOKE

```
GRANT tip_priv [(lista_coloane)][, tip_priv [(lista_coloane)] ...]
   ON {*.* | * | nume_db.* | nume_db.nume_tabel | nume_tabel}
   FROM utilizator [, utilizator] ...
```

Această instrucțiune revocă privilegiile utilizatorului sau utilizatorilor specificați. Clauzele *tip_priv*, *lista_coloane* și *utilizator* sunt specificate în același mod ca în instrucțiunea GRANT. Aceleași categorii de specificatori sunt permise și pentru clauza ON.

Instrucțiunea REVOKE nu elimină utilizatorul din tabelul de acordare user. Aceasta înseamnă că utilizatorul poate în continuare să se conecteze la serverul MySQL. Pentru a elimina în totalitate utilizatorul, trebuie să ștergeți manual intrările aferente acestuia din tabelele de acordare.

Instrucțiunea REVOKE a fost introdusă în versiunea MySQL 3.22.11.

Revocați toate privilegiile utilizatorului mare_si_tare@localhost:

```
REVOKE ALL ON *.* mare_si_tare@localhost
```

Revocați privilegiile care permit utilizatorului member_mgr să modifice tabelul membru din baza de date samp_db:

```
REVOKE INSERT,DELETE,UPDATE ON samp_db.membru
   FROM member_mgr@boa.snake.net
```


Revocați toate privilegiile pentru un singur tabel din baza de date menagerie aferente utilizatorului anonim care se conectează de la gazda locală:

```
REVOKE ALL ON menagerie.pet FROM ""@localhost
```

SELECT

```
SELECT
   [optiuni_selectare]
   lista_selectare
   [INTO OUTFILE 'nume_fisier' optiuni_export]
   [FROM lista_tabele
   [WHERE expresie_unde]
   [GROUP BY lista_coloane]
   [HAVING expresie_unde]
   [ORDER BY {unsigned_integer | nume_coloana | formula}[ASC | DESC] , ...]
   [LIMIT [offset,] randuri]
   [PROCEDURE nume_procedura ]
```

Regăsește rânduri din tabelele specificate. De fapt, toate clauzele sunt opționale, în afară de cuvântul cheie SELECT și de clauza *lista_selectare*, care acceptă instrucțiuni ce pur și simplu evaluează expresii:

```
SELECT "unu plus unu =", 1+1
```

Clauza optiuni_selectare, dacă există, poate conține următoarele opțiuni:

- ALL

  DISTINCT

  DISTINCTROW

  Aceste cuvinte cheie controlează dacă vor fi returnate sau nu rânduri duplicate. ALL determină returnarea tuturor rândurilor. Aceasta este opțiunea prestabilită. DISTINCT și DISTINCTROW specifică eliminarea rândurilor duplicate din setul de rezultate.

- HIGH_PRIORITY

  Prin specificarea acestei opțiuni, instrucțiunea dobândește o prioritate mai ridicată, dacă în mod normal ar fi trebuit să aștepte. Dacă alte instrucțiuni, precum INSERT și UPDATE, așteaptă să scrie în tabele specificate în instrucțiunea SELECT deoarece un alt client citește tabelele, opțiunea HIGH_PRIORITY face astfel încât instrucțiunii SELECT să i se acorde prioritate față de aceste instrucțiuni de scriere. Această opțiune trebuie folosită numai pentru instrucțiuni SELECT care se execută rapid și trebuie executată imediat, deoarece reduce viteza de execuție a instrucțiunilor de scriere. Opțiunea HIGH_PRIORITY a fost introdusă în MySQL 3.22.9.

- SQL_BIG_RESULT

  SQL_SMALL_RESULT

  Aceste cuvinte cheie arată că setul de rezultate va fi mare sau mic, informații pe care utilitarul de optimizare le poate folosi pentru a prelucra interogarea într-o manieră mai eficientă. Opțiunea SQL_SMALL_RESULT a fost introdusă în versiunea MySQL 3.22.12. Opțiunea SQL_BIG_RESULT a fost introdusă în versiunea MySQL 3.23.

- STRAIGHT_JOIN

  Forțează executarea unirii tabelelor în ordinea specificată în clauza FROM. Se poate folosi dacă credeți că utilitarul de optimizare nu face alegerea cea mai bună.

Clauza *lista_selectare* precizează coloanele de ieșire care trebuie returnate. Numele coloanelor, dacă sunt mai multe, trebuie separate prin virgule. Coloanele pot fi referințe la coloane din tabel sau expresii. Orice coloană poate primi un alias folosind sintaxa AS *nume_alias*. Apoi, aliasul devine numele coloanei în datele de ieșire și se poate face referire la acesta în clauzele GROUP BY, ORDER BY și HAVING. Totuși, nu puteți face referire la un alias într-o clauză WHERE.

Notația specială * are semnificația „toate coloanele din tabelele denumite în clauza FROM", iar *nume_tabel*.* este echivalentă cu „toate coloanele din tabelul denumit".

Rezultatul unei instrucțiuni SELECT poate fi scris într-un fișier *nume_fisier* folosind o clauză INTO '*nume_fisier*' OUTFILE. Trebuie să dispuneți de privilegiul FILE, iar fișierul nu trebuie să existe. Numele fișierului este interpretat folosind aceleași reguli care se aplică la citirea fișierelor non-LOCAL în instrucțiunea LOAD DATA. Sintaxa clauzei *optiuni_export* este aceeași ca a clauzei optiuni_import din instrucțiunea LOAD DATA. Pentru mai multe informații, consultați prezentarea instrucțiunii LOAD DATA.

Clauza FROM denumește unul sau mai multe tabele din care trebuie selectate rândurile. MySQL acceptă utilizarea următoarelor tipuri JOIN în instrucțiunile SELECT:

```
nume_tabele
    lista_tabele, nume_tabel
    lista_tabele [CROSS] JOIN nume_tabel
    lista_tabele INNER JOIN nume_tabel
    lista_tabele STRAIGHT_JOIN nume_tabel
    lista_tabele LEFT [OUTER] JOIN nume_tabel ON expr_conditionala
    lista_tabele LEFT [OUTER] JOIN nume_tabel USING (lista_coloane)
    lista_tabele NATURAL LEFT [OUTER] JOIN nume_tabel
    { oj lista_tabele LEFT OUTER JOIN nume_tabel ON expr_conditionala }
```

Tipurile de uniri selectează rânduri din tabelele denumite așa cum se va vedea în descrierile de mai jos, deși rândurile efectiv returnate clientului pot fi limitate de clauzele WHERE, HAVING sau LIMIT.

Pentru un singur tabel denumit, instrucțiunea SELECT regăsește rânduri din tabel.

Dacă sunt specificate mai multe tabele separate prin virgulă, instrucțiunea SELECT returnează toate combinațiile posibile de rânduri din tabele. Utilizarea clauzelor JOIN, CROSS JOIN sau INNER JOIN este similară cu utilizarea virgulelor. Clauza STRAIGHT_JOIN este similară, dar forțează utilitarul de optimizare să unească tabelele în ordinea în care sunt specificate. Această clauză se poate folosi dacă credeți că utilitarul de optimizare nu face alegerea cea mai bună.

Clauza LEFT JOIN regăsește rânduri din tabelele unite, dar forțează generarea unui rând pentru fiecare rând din tabelul din stânga, chiar dacă nu există nici un rând echivalent în tabelul din dreapta. Când nu există nici o corespondență, coloanele din tabelul din dreapta sunt returnate sub formă de valori NULL. Rândurile echivalente sunt determinate

conform condiției specificate în clauza ON expr_conditionala sau în clauza USING (lista_coloane). expr_conditionala este o expresie de forma care poate fi folosită în clauza WHERE. lista_coloane constă din una sau mai multe nume de coloane separate prin virgulă, iar toate aceste coloane trebuie să se regăsească în ambele tabele unite. LEFT OUTER JOIN este echivalentă cu LEFT JOIN, așa cum este și sintaxa care începe cu o (inclusă pentru compatibilitatea cu ODBC).

NATURAL LEFT JOIN este echivalentă cu LEFT JOIN USING (*lista_coloane*), unde *lista_coloane* denumește fiecare coloană comună pentru ambele tabele.

Tabelele pot primi aliasuri în clauza FROM, folosindu-se sintaxa *nume_tabel nume_alias* sau *nume_tabel* AS *nume_alias*. Referințele la o coloană cu alias inserate în altă parte a instrucțiunii SELECT trebuie să folosească aliasul respectiv.

Clauza WHERE specifică o expresie aplicată rândurilor selectate din tabelele denumite în clauza FROM. Rândurile care nu satisfac criteriile date de expresie sunt eliminate. Setul de rezultate poate fi restrâns și mai mult, prin intermediul clauzelor HAVING și LIMIT.

Clauza GROUP BY lista_coloane grupează rânduri din setul de rezultate conform coloanelor denumite în listă. Această clauză este folosită la specificarea unor funcții de sumar precum COUNT() sau MAX() în clauza lista_selectie. În lista lista_selectie se poate face referire folosind nume de coloane sau aliasuri, respectiv poziția acestora în lista de selecție. Pozițiile coloanelor sunt numerotate începând de la 1.

Clauza HAVING specifică o expresie secundară, folosită pentru a limita rândurile după ce acestea au satisfăcut condițiile denumite de clauza WHERE. Rândurile care nu satisfac condiția HAVING sunt respinse. Clauza HAVING este utilă pentru expresii care implică funcții de sumar, ce nu pot fi testate în clauza WHERE. Totuși, dacă o condiție este admisă în clauza WHERE sau în clauza HAVING, este preferabil ca aceasta să fie plasată în clauza WHERE, unde va fi supusă analizei utilitarului de optimizare.

Clauza ORDER BY indică modul de sortare a setului de rezultate. Ca și la GROUP BY, coloanele pot fi desemnate prin nume sau prin alias, respectiv prin poziția în cadrul listei de selecție a coloanelor. În mod prestabilit, coloanele de ieșire sunt sortate în ordine ascendentă. Pentru a specifica în mod explicit o ordine de sortare pentru o coloană, inserați expresia ASC (ascending – crescător) sau DESC (descending – descrescător). Începând de la versiunea MySQL 3.23.2, mai puteți specifica și o expresie drept coloană de sortare. De exemplu, ORDER BY RAND() returnează rândurile în ordine aleatoare.

Clauza LIMIT se poate folosi pentru a selecta un număr de rânduri din setul de rezultate. LIMIT n returnează primele n rânduri. LIMIT m,n returnează n rânduri, începând de la rândul m. Pentru LIMIT, rândurile sunt numerotate începând de la 0, nu de la 1.

PROCEDURE denumește o procedură căreia îi vor fi trimise datele din setul de rezultate înainte de a i se returna clientului un set de rezultate. Începând de la versiunea MySQL 3.23, puteți folosi PROCEDURE ANALYSE() pentru a obține informații despre caracteristicile datelor din coloanele denumite în lista de selecție a coloanelor.

Următoarele instrucțiuni prezintă unele moduri de utilizare a instrucțiunii SELECT. Pentru mai multe exemple, consultați capitolul 1, „Introducere în MySQL și SQL".

Selectează întregul conținut al unui tabel:

```
SELECT * FROM presedinte
```

Selectează întregul conținut, dar sortează după nume:

```
SELECT * FROM presedinte ORDER BY nume, prenume
```

Selectează înregistrările președinților născuți după data de "1900-01-01":

```
SELECT * FROM presedinte WHERE nastere = "1900-01-01"
```

Aceeași operație, dar se execută sortarea după data nașterii:

```
SELECT * FROM presedinte WHERE nastere = "1900-01-01" ORDER BY nastere
```

Determină care sunt statele reprezentate prin rânduri în tabelul membru:

```
SELECT DISTINCT stat FROM membru;
```

Selectează rânduri din tabelul membru și scrie coloanele sub formă de valori separate prin virgulă într-un fișier:

```
SELECT * INTO OUTFILE "/tmp/membru.txt"
   FIELDS TERMINATED BY "," FROM membru
```

Selectează primele cinci punctaje obținute la un anumit eveniment:

```
SELECT * FROM puncte WHERE eveniment_id = 9 ORDER BY puncte DESC LIMIT 5
```

SET

```
SET [OPTION] stabilire_optiuni, ...
```

Instrucțiunea SET este folosită pentru a specifica o varietate de opțiuni. Cel mai indicat este să omiteți cuvântul OPTION, deoarece acesta va fi eliminat într-o versiune ulterioară a sistemului MySQL.

Valorile stabilire_optiuni pot fi oricare dintre articolele prezentate în lista următoare:

- CHARACTER SET {*nume_setcaractere* | DEFAULT}

  Specifică setul de caractere folosit de către client. Șirurile trimise clientului, respectiv cele primite de la client, sunt mapate folosind acest set de caractere. Singurul set de caractere disponibil în prezent este cp1251_koi18. Numele setului de caractere DEFAULT restabilește setul de caractere prestabilit:

  ```
  SET CHARACTER SET cp1251_koi18
  SET CHARACTER SET DEFAULT
  ```

- INSERT_ID = *n*

  Specifică valoarea care va fi folosită în următoarea instrucțiune INSERT la inserția unei coloane AUTO_INCREMENT. Aceasta este folosită pentru prelucrarea jurnalului de actualizare.

  ```
  SET INSERT_ID = 1973
  ```

- LAST_INSERT_ID = *n*

  Specifică valoarea care va fi returnată de LAST_INSERT_ID(). Aceasta este folosită pentru prelucrarea jurnalului de actualizare.

  ```
  SET LAST_INSERT_ID = 48731
  ```

- PASSWORD [FOR *utilizator*] = PASSWORD("*parola*")

  În lipsa clauzei FOR, stabilește parola utilizatorului curent ca fiind "*parola*". Dacă această clauză există, stabilește parola pentru utilizatorul dat. Pentru a putea configura parola unui alt utilizator, trebuie să aveți privilegii pentru modificarea bazei de date mysql. *utilizator* este specificat în forma *nume_utilizator@nume_gazda*, folosind aceleași tipuri de valori pentru *nume_utilizator* și *nume_gazda* acceptabile pentru instrucțiunea GRANT.

  ```
  SET PASSWORD = PASSWORD("secret")
  SET PASSWORD FOR paul = PASSWORD("secret")
  SET PASSWORD FOR paul@localhost = PASSWORD("secret")
  SET PASSWORD FOR bill@%.bigcorp.com = PASSWORD("old-sneep")
  ```


- SQL_AUTO_IS_NULL = {0 | 1}

  Dacă această opțiune are valoarea 1, ultimul rând conținând o valoare AUTO_INCREMENT care a fost inserat poate fi selectat folosind o clauză WHERE de forma WHERE *auto_inc_col* IS NULL. Această clauză este folosită de unele programe ODBC, precum Access. Valoarea prestabilită este 1. Această opțiune a fost introdusă în MySQL 3.23.5.

  ```
  SET_SQL_AUTO_IS_NULL = 0
  ```

- SQL_BIG_SELECTS = {0 | 1}

  Instrucțiunile SELECT care vor returna mai mult de max_join_size rânduri sunt permise dacă această opțiune are valoarea 1. În caz contrar, interogarea va eșua. Valoarea prestabilită este 1.

  ```
  SET_SQL_BIG_SELECTS = 0
  ```

- SQL_BIG_TABLES = {0 | 1}

  Dacă această opțiune are valoarea 1, toate tabelele temporare sunt stocate pe disc, nu în memorie. Performanțele sunt mai lente, dar instrucțiunile SELECT care necesită tabele temporare mari nu vor genera erori de tip „tabele pline". Valoarea prestabilită este 0 (stocarea tabelelor temporare în memorie). În mod normal, această opțiune nu este necesară pentru MySQL 3.23 și versiunile ulterioare.

  ```
  SET_SQL_BIG_TABLES = 1
  ```

- SQL_LOG_OFF = {0 | 1}

  Dacă această opțiune are valoarea 1, interogările clientului curent nu sunt consemnate în fișierul jurnal general. Dacă opțiunea are valoarea 0, este activată consemnarea în jurnal pentru clientul respectiv. Clientul trebuie să aibă privilegiul PROCESS pentru ca această instrucțiune să aibă efect. Procesul de consemnare în jurnalul de actualizare nu este afectat.

  ```
  SET SQL_LOG_OFF = 1
  ```

- SQL_LOG_UPDATE = {0 | 1}

  Este similară cu instrucțiunea SQL_LOG_OFF (inclusiv necesitatea privilegiului PROCESS), cu excepția faptului că afectează consemnarea în jurnalul de actualizare și nu în jurnalul general. SQL_LOG_UPDATE a fost introdusă în MySQL versiunea 3.22.5.

  ```
  SET SQL_LOG_UPDATE = 1
  ```

- SQL_LOW_PRIORITY_UPDATES = {0 | 1}

  Dacă această opțiune are valoarea 1, instrucțiunile care aduc modificări conținutului unui tabel (DELETE, INSERT, REPLACE, UPDATE) așteaptă până când nici o instrucțiune SELECT pentru tabelul respectiv nu mai este activă sau suspendată. Instrucțiunile

SELECT care sosesc în timp ce o altă instrucțiune este activă încep să se execute imediat, în loc să aștepte instrucțiunile de modificare cu prioritate redusă.

Opțiunea SQL_LOW_PRIORITY_UPDATES a fost introdusă în MySQL 3.22.5.

```
SET SQL_LOW_PRIORITY_UPDATES = 0
```

- SQL_SELECT_LIMIT = {*n* | DEFAULT}

Specifică numărul maxim de înregistrări ce vor fi returnate de către o instrucțiune SELECT. Prezența unei clauze LIMIT explicite într-o instrucțiune are prioritate în raport cu această opțiune. Valoarea prestabilită este „fără limite". Valoarea DEFAULT restaurează valoarea prestabilită, dacă ați modificat-o.

```
SET SQL_SELECT_LIMIT = 100000
SET SQL_SELECT_LIMIT = DEFAULT
```

- SQL_WARNINGS = {0 | 1}

Dacă această opțiune este configurată la valoarea 1, MySQL raportează numărul de avertizări chiar și pentru inserția unui singur rând. În mod normal, numerele de avertizări sunt raportate numai pentru instrucțiunile INSERT care inserează mai multe rânduri. Valoarea prestabilită este 0. Opțiunea SQL_WARNINGS a fost introdusă în versiunea MySQL 3.22.11.

```
SET SQL_WARNINGS = 1
```

- TIMESTAMP = {*valoare_amprenta* | DEFAULT}

Specifică o valoare TIMESTAMP. Este folosită pentru prelucrarea jurnalului de actualizare.

```
SET TIMESTAMP = DEFAULT
```

SHOW

```
SHOW COLUMNS FROM nume_tabel [FROM nume_db] [LIKE model]
SHOW DATABASES [LIKE model]
SHOW GRANTS FOR nume_utilizator
SHOW INDEX FROM nume_tabel [FROM nume_db]
SHOW PROCESSLIST
SHOW STATUS
SHOW TABLE STATUS [FROM nume_db][LIKE model]
SHOW TABLES [FROM nume_db] [LIKE model]
SHOW VARIABLES [LIKE model]
```

Diferitele forme ale instrucțiunii SHOW furnizează informații despre baze de date, tabele, coloane și indexuri, respectiv informații despre modul de operare a serverului. Numeroase forme acceptă o clauză opțională FROM *nume_db*, permițându-vă să specificați baza de date ale cărei informații trebuie prezentate. În cazul în care clauza nu este prezentă, se folosește baza de date prestabilită. Unele forme permit o clauză opțională LIKE *model* pentru limitarea ieșirii la valorile care corespund modelului. Modelul trebuie să fie un șir sau un model SQL care să conțină caracterele % sau _.

SHOW COLUMNS

Instrucțiunea SHOW COLUMNS enumeră coloanele pentru tabelul dat. SHOW FIELDS este sinonimă cu instrucțiunea SHOW COLUMNS.

```
SHOW COLUMNS FROM presedinte
SHOW FIELDS FROM presedinte
SHOW COLUMNS FROM presedinte FROM samp_db
SHOW COLUMNS FROM tables_priv FROM mysql LIKE "%priv".
```

Datele de ieșire ale instrucțiunii SHOW COLUMNS conțin următoarele coloane:

- Field

  Numele coloanei din tabel.
- Type

  Tipul coloanei.
- Null

  Dacă coloana poate sau nu să conțină valori NULL. Poate fi YES sau blanc (nu).
- Key

  Dacă coloana este indexată sau nu.
- Default

  Valoarea prestabilită a coloanei.
- Extra

  Informații suplimentare despre coloană.
- Privileges

  Privilegiile aferente coloanei. Aceste informații sunt prezentate numai începând de la versiunea MySQL 3.23.

SHOW DATABASES

Instrucțiunea SHOW DATABASES enumeră bazele de date disponibile pe gazda serverului.

```
SHOW DATABASES
SHOW DATABASES LIKE "test%"
```

SHOW GRANTS

Instrucțiunea SHOW GRANTS afișează informațiile de acordare cu privire la utilizatorul specificat, care trebuie specificate sub forma *nume_utilizator@nume_gazda*, folosind aceleași tipuri de valori pentru *nume_utilizator* și *nume_gazda* care sunt acceptabile pentru instrucțiunea GRANT. Instrucțiunea SHOW GRANTS a fost introdusă în MySQL versiunea 3.23.4.

```
SHOW GRANTS FOR root@localhost
SHOW GRANTS FOR ""@pit-viper.snake.net
```

SHOW INDEX

Instrucțiunea SHOW INDEX afișează informații despre indexurile unui tabel. Instrucțiunea SHOW KEYS este sinonimă cu SHOW INDEX.

```
SHOW INDEX FROM puncte
SHOW KEYS FROM puncte
SHOW INDEX FROM samp_db.puncte
SHOW INDEX FROM puncte FROM samp_db
```

Datele de ieșire ale funcției `SHOW INDEX` conțin următoarele coloane:

- `Table`

  Numele tabelului care conține indexul.
- `Non_unique`

  Această valoare este 1 dacă indexul poate conține valori duplicate, respectiv 0 dacă indexul nu poate conține asemenea valori.
- `Key_name`

  Numele indexului.
- `Seq_in_index`

  Numărul coloanei din index. Coloanele sunt numerotate începând de la 1.
- `Column_name`

  Numele coloanei.
- `Collation`

  Ordinea de sortare a coloanelor în cadrul indexului. Valorile pot fi `A` (ascending - crescător), `D` (descending - descrescător) sau `NULL` (fără sortare). Cheile descrescătoare nu sunt încă disponibile, dar vor fi implementate în viitor.
- `Cardinality`

  Numărul de valori unice din index. Această valoare este actualizată prin rularea programelor `myisamchk` sau `isamchk` cu opțiunea `--analyze`.
- `Sub_part`

  Lungimea prefixului, dacă este indexat numai un prefix al coloanei. Are valoarea `NULL` dacă este indexată întreaga coloană.
- `Packed`

  Modul în care este împachetată cheia, respectiv `NULL` dacă nu este împachetată. Această coloană este prezentată numai începând de la versiunea 3.23.
- `Comment`

  Rezervat pentru comentarii interne referitoare la index. Această coloană este prezentată numai pentru MySQL 3.23 și versiunile ulterioare.

SHOW PROCESSLIST

Funcția `SHOW PROCESSLIST` afișează informații despre firele care se execută în cadrul serverului. Datele de ieșire conțin următoarele coloane:

- `Id`

  Numărul de identificare a firului pentru client.
- `User`

  Numele de client asociat cu firul.

- `Host`

  Gazda de la care este conectat clientul.
- `db`

  Baza de date prestabilită pentru fir.
- `Command`

  Comanda executată de fir.
- `Time`

  Intervalul de timp folosit de comanda aflată în execuție în fir, exprimat în secunde.
- `State`

  Informații despre acțiunile sistemului MySQL în timpul prelucrării unei instrucțiun SQL. Această valoare poate fi utilă pentru semnalarea unei probleme a sistemulu MySQL sau la inserția în lista de corespondență pentru MySQL a unei întrebări privinc motivul pentru care firul rămâne într-o anumită stare pentru o perioadă de timp ma îndelungată.
- `Info`

  Interogarea care se execută.

SHOW STATUS

Instrucțiunea `SHOW STATUS` afișează variabilele de stare ale serverului și valorile acestora

- `Aborted_clients`

  Numărul de conexiuni client întrerupte datorită clienților care nu au închis conexiunea în mod adecvat.
- `Aborted_connects`

  Numărul de încercări ratate de conectare la server.
- `Connections`

  Numărul de încercări de conectare la server (atât reușite, cât și nereușite). Dacă această opțiune are o valoare foarte mare, puteți folosi conexiuni persistente pentru clienții dumneavoastră, dacă este posibil.
- `Created_tmp_tables`

  Numărul tabelelor temporare create în timpul prelucrării interogărilor.
- `Delayed_errors`

  Numărul de erori care s-au produs în timpul prelucrării rândurilor `INSERT DELAYED`.
- `Delayed_insert_threads`

  Numărul curent de rutine de tratare `INSERT_DELAYED`.
- `Delayed_writes`

  Numărul de rânduri `INSERT DELAYED` care au fost scrise.
- `Flush_commands`

  Numărul de comenzi `FLUSH` care au fost executate.

- `Handler_delete`
  Numărul de cereri de ștergere a unui rând dintr-un tabel.
- `Handler_read_first`
  Numărul de cereri de citire a primului rând dintr-un tabel.
- `Handler_read_key`
  Numărul de cereri de citire a unui rând în funcție de valoarea dintr-un index.
- `Handler_read_next`
  Numărul de cereri de citire a rândului următor din ordinea de indexare.
- `Handler_read_rnd`
  Numărul de cereri de citire a unui rând în funcție de poziția rândului. Dacă acest număr este cu mult mai mare decât toate celelalte numere raportate, fie executați multe interogări care impun baleierea integrală a tabelului, fie nu folosiți indexurile în mod adecvat.
- `Handler_update`
  Numărul de cereri de actualizare a unui rând dintr-un tabel.
- `Handler_write`
  Numărul de cereri de inserție a unui rând într-un tabel.
- `Key_blocks_used`
  Numărul de blocuri care vor fi utilizate în zona cache a indexului.
- `Key_read_requests`
  Numărul de cereri de citire a unui bloc din zona cache a indexului.
- `Key_reads`
  Numărul de citiri fizice de blocuri de index de pe disc.
- `Key_write_requests`
  Numărul de cereri de scriere a unui bloc în zona cache a indexului.
- `Key_writes`
  Numărul de scrieri fizice de blocuri de index pe disc.
- `Max_used_connections`
  Numărul maxim de conexiuni care au fost deschise simultan.
- `Not_flushed_delayed_rows`
  Numărul rândurilor care așteaptă să fie scrise pentru interogările `INSERT DELAYED`.
- `Not_flushed_key_blocks`
  Numărul blocurilor din zona cache a cheilor care au fost modificate, dar care nu au fost încă evacuate pe disc.
- `Open_files`
  Numărul de fișiere deschise.

- `Open_streams`
  Numărul fluxurilor deschise. Un flux este un fișier deschis cu `fopen()`; acest procede se aplică numai fișierelor jurnal.
- `Open_tables`
  Numărul tabelelor deschise.
- `Opened_tables`
  Numărul total al tabelelor care au fost deschise. Dacă acest număr are o valoare mar se recomandă mărirea dimensiunii zonei cache a tabelului.
- `Questions`
  Numărul interogărilor recepționate de server (include atât interogările reușite, cât cele nereușite).
- `Slow_queries`
  Numărul interogărilor care necesită un timp de execuție mai lung deci `long_query_time`.
- `Threads_connected`
  Numărul de conexiuni deschise la un moment dat.
- `Threads_running`
  Numărul firelor care nu sunt inactive.
- `Uptime`
  Intervalul de timp (în secunde) scurs de la momentul pornirii serverului.

SHOW TABLE STATUS

Instrucțiunea `SHOW TABLE STATUS` afișează informații cu caracter descriptiv cu privire la ta belele dintr-o bază de date. `SHOW TABLE STATUS` a fost introdusă în versiunea MySQL 3.23

```
SHOW TABLE STATUS
SHOW TABLE STATUS FROM samp_db
SHOW TABLE STATUS FROM mysql LIKE "%priv"
```

Datele de ieșire ale instrucțiunii `SHOW TABLE STATUS` conțin următoarele coloane:

- `Name`
  Numele tabelului.
- `Type`
  Tipul tabelului; acesta poate fi NISAM (ISAM), MyISAM sau HEAP.
- `Row_format`
  Formatul de stocare a rândurilor; acesta poate fi `Fixed` (rânduri de lungime fixă) `Dynamic` (rânduri de lungime variabilă) sau `Compressed`.
- `Rows`
  Numărul rândurilor dintr-un tabel.
- `Avg_row_length`
  Numărul mediu de octeți folosit de rândurile tabelului.

- `Data_length`
  Dimensiunea efectivă (în octeți) a fișierului de date al tabelului.
- `Max_data_length`
  Dimensiunea maximă (în octeți) la care poate ajunge fișierul de date al tabelului.
- `Index_length`
  Dimensiunea efectivă (în octeți) a fișierului index.
- `Data_free`
  Numărul de octeți nefolosiți din fișierul de date. Dacă acest număr are o valoare foarte mare, se recomandă emiterea unei instrucțiuni `OPTIMIZE TABLE` pentru tabel.
- `Auto_increment`
  Următoarea valoare care va fi generată pentru o coloană `AUTO_INCREMENT`.
- `Create_time`
  Momentul creării tabelului.
- `Update_time`
  Momentul celei mai recente modificări a tabelului.
- `Check_time`
  Momentul ultimei verificări sau reparări a tabelului cu ajutorul utilitarului `myisamchk`; valoarea este `NULL` dacă tabelul nu a fost niciodată verificat sau reparat.
- `Create_options`
  Opțiunile suplimentare care au fost specificate în clauza `optiuni_tabel` a instrucțiunii `CREATE TABLE` care a creat tabelul.
- `Comment`
  Textul oricărui comentariu specificat la crearea tabelului.

SHOW TABLES

Instrucțiunea `SHOW TABLES` afișează o listă a tabelelor dintr-o bază de date.

```
SHOW TABLES
SHOW TABLES FROM samp_db
SHOW TABLES FROM mysql LIKE "%priv"
```

SHOW VARIABLES

Instrucțiunea `SHOW VARIABLES` afișează o listă a variabilelor serverului și a valorilor acestora. Variabilele serverului sunt descrise în prezentarea programului `mysqld` din Anexa E, „Referință de programe MySQL".

```
SHOW VARIABLES
SHOW VARIABLES LIKE "%thread"
```

UNLOCK TABLES

```
UNLOCK TABLES
```

Această instrucțiune anulează toate blocările de tabele deținute de clientul curent. Instrucțiunea `UNLOCK TABLE` este sinonimă cu `UNLOCK TABLES`.

UPDATE

```
UPDATE [LOW_PRIORITY] nume_tabel
  SET nume_coloana=expr [, nume_coloana=expr ] ...
  [WHERE unde_expr] [LIMIT n]
```

Aduce modificări conținutului rândurilor existente în tabelul `nume_tabel`. Rândurile de modificat sunt cele selectate prin expresia specificată în clauza `WHERE`. Pentru rândurile selectate, fiecare coloană denumită în clauza `SET` primește valoarea expresiei corespunzătoare.

```
UPDATE membru SET expirare = NULL, telefon = "197-602-4832"
  WHERE membru_id = 14
```

Dacă nu este dată nici o clauză `WHERE`, *toate rândurile tabelului vor fi actualizate.*

Instrucțiunea `UPDATE` returnează numărul rândurilor care au fost actualizate. Cu toate acestea un rând nu este considerat ca fiind actualizat decât atunci când se modifică efectiv valoarea unei coloane. A atribui unei coloane o valoare pe care aceasta o conține deja nu se consideră ca afectează rândul. Dacă aplicația dumneavoastră *trebuie* să cunoască numărul rândurilor care au corespuns clauzei `WHERE` indiferent dacă instrucțiunea `UPDATE` a modificat vreo valoare, trebuie să specificați indicatorul `CLIENT_FOUND_ROWS` atunci când stabiliți o conexiune cu serverul. Consultați prezentarea funcției `mysql_real_connect()` în Anexa F, „Referință API C".

Clauza `LOW_PRIORITY` determină amânarea execuției instrucțiunii până când nici un client nu mai citește din tabel. Această clauză a fost introdusă în MySQL 3.22.5.

Dacă este dată clauza `LIMIT`, valoarea n specifică numărul maxim de rânduri care vor fi actualizate. Clauza `LIMIT` a fost introdusă în MySQL 3.23.3.

USE

```
USE nume_db
```

Selectează *nume_db* pentru a o transforma în bază de date curentă (baza de date prestabilită pentru referirile la tabele care nu includ un nume explicit al unei baze de date). Instrucțiunea `USE` eșuează dacă baza de date nu există sau dacă nu aveți nici un privilegiu pentru a obține accesul la aceasta.

Sintaxa comentariilor

Această secțiune descrie modul de scriere a comentariilor în programul dumneavoastră SQL. De asemenea, semnalează un neajuns al programului client `mysql` în raport cu interpretarea comentariilor.

Comentariile sunt frecvent folosite în fișierele de interogare care sunt executate folosind `mysql` în modul batch, deci trebuie să fiți foarte atent la această limitare atunci când scrieți asemenea fișiere.

Serverul MySQL înțelege trei tipuri de comentarii:

- Toate caracterele cuprinse între `#` și sfârșitul liniei sunt tratate drept comentariu. Această sintaxă este similară celei folosite în majoritatea interpretoarelor și în Perl.
- Toate caracterele cuprinse între `/*` și `*/` sunt tratate drept comentariu. Această formă de comentarii se poate extinde pe mai multe linii. Această sintaxă este similară celei folosite în limbajul C.

- Începând de la versiunea MySQL 3.23.3, un comentariu poate începe folosind caracterele „-- " (două cratime și un spațiu); toate caracterele cuprinse între cratime și sfârșitul liniei sunt tratate drept comentariu. Acest stil de comentariu este folosit de alte sisteme de baze de date, cu excepția faptului că spațiul nu este obligatoriu. MySQL impune caracterul spațiu pentru eliminarea ambiguităților, astfel încât expresii precum *valoare1–valoare2*, unde *valoare2* este negativ, nu vor fi tratate drept comentarii.

La executarea interogărilor, serverul ignoră comentariile, cu excepția comentariilor de tip C care încep cu `/*!`, care se bucură de un tratament special. Textul comentariului trebuie să conțină cuvinte cheie din SQL, iar cuvintele cheie vor fi tratate de către serverul MySQL ca parte a instrucțiunii în care apare comentariul. De exemplu, următoarele linii sunt considerate de server ca fiind echivalente:

```
INSERT LOW_PRIORITY INTO tabelul_meu SET ...
INSERT /*! LOW_PRIORITY */ INTO tabelul_meu SET ...
```

Această formă de comentariu este destinată utilizării pentru extensii și cuvinte cheie specifice sistemului MySQL. MySQL va recunoaște cuvintele cheie, iar alte servere SQL le vor ignora. Astfel, este facilitată scrierea interogărilor care folosesc caracteristici specifice sistemului MySQL, dar care sunt funcționale și în alte sisteme de baze de date. Stilul de comentariu `/*!` a fost introdus în versiunea MySQL 3.22.7.

Începând de la versiunea MySQL 3.22.26, secvența `/*!` poate fi urmată de un număr de versiune, pentru a indica sistemului MySQL să ignore comentariul dacă numărul versiunii serverului este anterior versiunii respective. Comentariul din următoarea instrucțiune `UPDATE` este ignorat dacă versiunea serverului este anterioară versiunii 3.23.3:

```
UPDATE tabelul_meu SET coloana_mea = 100
   WHERE coloana_mea  100 /*!32303 LIMIT 100 */
```

Clientul `mysql` este mai limitat decât serverul MySQL în ceea ce privește capacitatea de a înțelege comentariile. `mysql` se încurcă dacă sunt inserate anumite construcții în comentariile scrise în stil C, deoarece folosește un analizor mai puțin sofisticat decât serverul. De exemplu, un caracter de citare care apare în interiorul unui comentariu scris în stil C va deruta programul, făcându-l să creadă că analizează un șir, și acesta va continua să caute sfârșitul șirului până la găsirea unui caracter de citare corespunzător. Instrucțiunile următoare ilustrează acest comportament:

```
mysql> SELECT /* I have no quote */ 1;
+---+
| 1 |
+---+
| 1 |
+---+

mysql> SELECT /* I've got a quote */ 1;
    '>
```

Clientul `mysql` analizează fără probleme prima instrucțiune, o trimite la server pentru execuție și afișează un alt prompt `mysql`. Cea de-a doua instrucțiune conține un comentariu cu un semn al citării fără pereche. Ca rezultat, `mysql` intră în modul de analiză a șirului și continuă să se găsească în modul respectiv și după ce ați introdus linia, așa cum se vede din promptul `'`. Pentru a ieși din această situație, tastați un caracter de citare corespunzător, urmat de o comandă `\c` pentru a anula interogarea. Pentru mai multe informații privind semnificația prompturilor clientului `mysql`, consultați Anexa E.

ANEXA E

Referință de programe MySQL

Această anexă descrie programele MySQL prezentate în lista următoare. Ulterior p parcursul acestei anexe, fiecare program este descris mai detaliat, fiind dată inclusiv scurtă descriere a rolului acestuia, a sintaxei de invocare, a opțiunilor pe care le accept și o descriere a variabilelor sale interne.

- `isamchk` și `myisamschk`

  Utilitare pentru verificarea și repararea tabelelor, care execută analiza distribuție cheilor, precum și dezactivarea și reactivarea indexurilor.

- `myisampack` și `isampack`

  Utilitare pentru generarea de tabele comprimate numai pentru citire.

- `mysql`

  Program interactiv, care dispune de caracteristici de editare a liniilor pentru expediere interogărilor către serverul MySQL; poate fi de asemenea folosit în modul batch, pen tru executarea interogărilor stocate într-un fișier.

- `mysql.server`

  Script pentru pornirea și oprirea serverului MySQL.

- `mysqlaccess`

  Script pentru testarea privilegiilor de acces.

- `mysqladmin`

  Utilitar pentru efectuarea de operații administrative.

- `mysqlbug`

  Script pentru generarea rapoartelor de semnalare a hibelor.

- `mysqld`

  Serverul MySQL; acest program trebuie să ruleze pentru a permite clienților să aib acces la bazele de date administrate de server.

- `mysqldump`

  Utilitar pentru transferul conținutului tabelelor dintr-o bază de date.

- `mysqlimport`

  Utilitar pentru încărcarea unor mari cantități de date în tabele.

- `mysqlshow`

  Utilitar care furnizează informații despre baze de date sau tabele.

- `safe_mysqld`

  Script pentru pornirea și monitorizarea serverului MySQL.

În descrierea sintaxei, informațiile opționale sunt indicate prin paranteze drepte ([ ]).

Specificarea opțiunilor de program

Majoritatea programelor MySQL acceptă numeroase opțiuni care influențează modul de operare a programului. Opțiunile pot fi specificate în linia de comandă sau în fișierele cu opțiuni. În plus, unele opțiuni pot fi specificate prin configurarea variabilelor de mediu. Opțiunile specificate în linia de comandă au prioritate în raport cu opțiunile specificate în orice alt mod, iar opțiunile din fișierele cu opțiuni au prioritate în raport cu valorile variabilelor de mediu.

Obținerea de asistență on-line de la programele MySQL

Fiecare program MySQL, cu excepția utilitarului `mysqlbug` și a scripturilor `mysql.server` și `safe_mysqld`, înțelege opțiunea `--help`. Această opțiune constituie o modalitate rapidă de a obține asistență on-line cu privire la un program chiar din partea programului respectiv. De exemplu, dacă nu știți să folosiți `mysqlimport`, invocați-l în modul următor, pentru a obține instrucțiuni:

```
% mysqlimport --help
```

Opțiunea `-?` este aceeași ca `help`, deși shell-ul dumneavoastră poate interpreta caracterul ? drept caracter de înlocuire pentru numele fișierelor:

```
% mysqlimport -?
mysqlimport: No match.
```

Dacă vi se întâmplă așa ceva, încercați următoarea comandă:

```
% mysqlimport -\?
```

Majoritatea opțiunilor au atât o formă scurtă (alcătuită dintr-o singură literă), cât și o formă lungă (cuvânt complet). Opțiunile `-?` și `--help` prezentate anterior reprezintă un exemplu în acest sens. Dacă o opțiune cu formă scurtă este urmată de o valoare, în majoritatea cazurilor opțiunea și valoarea pot fi separate prin spațiu. De exemplu, când specificați un nume de utilizator, `-upaul` este echivalent cu `-u paul`. Opțiunea `-p` (parola) este o excepție; orice valoare a parolei trebuie să urmeze imediat după litera `p`, fără nici un spațiu între ele.

Fiecare descriere de program prezintă toate opțiunile pe care un program le acceptă în acest moment. Dacă un program nu pare să recunoască o opțiune enumerată în descriere, s-ar putea să dispuneți de o versiune mai veche a programului, anterioară versiunii în care s-a adăugat opțiunea. (Verificați cu mare atenție sintaxa, pentru a vă asigura dacă nu cumva pur și simplu ați specificat incorect opțiunea respectivă.)

Opțiuni standard ale programelor MySQL

Numeroase opțiuni au o semnificație standard în mai multe programe MySQL. În loc de a le prezenta în descrierea fiecărui program care le acceptă, le vom enumera aici o singură dată, iar secțiunea „Opțiuni standard acceptate" din fiecare descriere de program

indică opțiunile pe care le acceptă fiecare program. (În secțiunea respectivă se vor prezenta numai numele în format lung, dar programele acceptă și opțiunile în forma scurt corespunzătoare, cu excepția situațiilor când se specifică altceva.)

Opțiunile standard sunt:

- `-C, --compress`

  Permite utilizarea comprimării pentru protocolul folosit la comunicația între client și server, dacă serverul o acceptă. Această opțiune este folosită numai de programele client. A fost introdusă în MySQL 3.22.3.

- `-# optiuni_depanare, --debug=optiuni_depanare`

  Activează datele de ieșire pentru depanare. Această opțiune are efect numai dacă MySQL a fost construit având inclus suportul pentru depanare. Șirul *`optiuni_depanare`* constă din opțiuni separate printr-un caracter două puncte. O valoare caracteristică este `d:t:o,`*`nume_fisier`*, care permite depanarea, activează introducerea apelurilor de funcții și urmărirea ieșirilor, și trimite datele de ieșire în fișierul *`nume_fisier`*.

  Dacă vă așteptați la o activitate intensă de depanare, trebuie să examinați fișierul `dbug/dbug.c` din distribuția sursă MySQL, pentru o descriere a tuturor opțiunilor pe care le puteți folosi.

- `--defaults-file=`*`nume_cale`*

  Precizează calea de acces spre un fișier cu opțiuni. În mod normal, programele caută fișierele cu opțiuni în numeroase locații, dar, dacă este specificată opțiunea `--defaults-file`, atunci va fi citit numai fișierul denumit. Dacă folosiți această opțiune, trebuie ca opțiunea respectivă să fie menționată prima în linia de comandă. Opțiunea a fost introdusă în versiunea MySQL 3.22.23.

- `-?,--`

  Indică programului să afișeze un mesaj de asistență și să-și încheie execuția.

- `-h` *`nume_gazda`*`, --host=`*`nume_gazda`*

  Specifică gazda la care se va stabili conexiunea (adică gazda unde rulează serverul). Această opțiune este folosită numai de către programele client.

- `-no-defaults`

  Dezactivează utilizarea oricăror fișiere cu opțiuni. Dacă folosiți opțiunea respectivă aceasta trebuie să fie prima opțiune din linia de comandă.

- `-p[`*`parola`*`],-password[=`*`parola`*`]`

  Specifică parola care se va folosi pentru conectarea la server. Dacă `parola` nu este specificată după numele opțiunii, programul vă va solicita să introduceți o parolă. Dacă `parola` este specificată, trebuie să urmeze imediat după numele opțiunii, fără spațiu între ele. Această opțiune este folosită numai de către programele client.

- `-W, --pipe`

  Folosește un canal denumit pentru conectarea la server. Această opțiune este folosită numai pentru programe client care rulează sub Windows.

- `-P numar_port, --port=numar_port`

  Pentru programele client, acesta este numărul portului pe care îl vor folosi acestea pentru a se conecta la server. Acesta este portul folosit pentru conexiunile TCP/IP (conexiuni unde gazda nu este `localhost`). Pentru `mysqld`, această opțiune specifică portul unde serverul va aștepta conexiunile TCP/IP.

- `--print-defaults`

  Afișează valorile opțiunilor care vor fi folosite dacă invocați programul fără opțiuni în linia de comandă. Opțiunea prezintă valorile care vor fi selectate din fișierele cu opțiuni (și din variabilele de mediu). `--print-defaults` este utilă pentru verificarea configurării adecvate a unui fișier cu opțiuni. Puteți folosi `--print-defaults` pentru a determina dacă opțiunile sunt preluate dintr-un fișier cu opțiuni. Dacă folosiți această opțiune, trebuie să fie menționată prima în linia de comandă.

- `-O var=valoare, --set-variable var=valoare`

  Numeroase programe au variabile (parametri de operare) pe care le puteți configura. Opțiunea `--set-variable` vă permite să configurați aceste variabile. *var* este numele variabilei, iar *valoare* este valoarea care va fi atribuită variabilei. Valorile sunt exprimate în octeți, pentru variabilele care reprezintă lungimi sau dimensiuni ale bufferelor. Valorile pot fi specificate sub formă de număr exact, respectiv ca număr cu sufixul `K` sau `M`, pentru a simboliza kiloocteți sau megaocteți. Sufixele nu sunt sensibile la diferența între majuscule și minuscule: `k` și `m` sunt echivalente cu `K`, respectiv `M`.

  Variabilele fiecărui program sunt prezentate în descrierea programului din această anexă și sunt de asemenea afișate atunci când invocați programul cu opțiunea `--help`.

- `-s, --silent`

  Specifică modul silențios. Aceasta nu înseamnă în mod obligatoriu că programul este complet silențios, ci numai că furnizează mai puține date de ieșire ca de obicei. Mai multe programe permit specificarea acestei opțiuni de mai multe ori, pentru a determina programul să devină tot mai silențios (operația se poate executa și în fișierele cu opțiuni).

- `-S nume_cale, --socket=nume_cale`

  Pentru programele client, aceasta este numele căii de acces complete a fișierului soclu care se va folosi pentru conectarea la server. Pentru conexiunile cu `localhost` se va folosi un fișier soclu.

- `-u nume_utilizator --user=nume_utilizator`

  Pentru programe client, acesta este numele de utilizator MySQL care se va folosi pentru conectarea la server. Dacă această opțiune nu este specificată, valoarea prestabilită este numele dumneavoastră de conectare sub UNIX, respectiv `ODBC` sub Windows. Pentru `mysqld`, această opțiune indică numele contului UNIX care se va folosi pentru rularea serverului. (Serverul trebuie lansat sub contul `root` pentru ca această opțiune să fie utilizabilă.)

- `-v, --verbose`

  Specifică modul amănunțit; programul produce mai multe date de ieșire decât de obicei. Numeroase programe permit specificarea acestei opțiuni de mai multe ori, pentru a determina programul să devină tot mai „vorbăreț" (operația se poate executa și în fișierele cu opțiuni).

- `-V, --version`

  Indică programului să afișeze șirul cu informații despre versiune și să-și încheie execuția.

Fișiere cu opțiuni

Fișierele cu opțiuni au fost introduse în MySQL 3.22. Acestea vă permit să stocați opțiuni de program MySQL, astfel încât să nu fie necesar să le tastați în linia de comandă de fiecare dată când invocați un program. Fișierele cu opțiuni sunt citite de către programele `isamchk`, `myisamchk`, `myisampack`, `mysql`, `mysqladmin`, `mysqld`, `mysqldump`, `mysqlimport`, `mysql.server` și `pack_isam`. Puteți găsi un exemplu de fișier cu opțiuni, denumit `my-example.cnf`, în catalogul `share/mysql` din catalogul de instalare MySQL sau în catalogul `support-files` al unei distribuții sursă.

Orice opțiune specificată într-un fișier cu opțiuni poate fi anulată prin specificarea explicită în linia de comandă a unei valori diferite a opțiunii.

Programele MySQL consultă numeroase fișiere în căutarea opțiunilor; totuși, nu este o eroare dacă un fișier cu opțiuni lipsește. Sub UNIX, sunt explorate în căutarea opțiunilor următoarele fișiere, în ordinea dată:

Numele fișierului	Conținut
`/etc/my.cnf`	Opțiuni globale
DATADIR`/my.cnf`	Opțiuni specifice serverului
`~/.my.cnf`	Opțiuni specifice utilizatorului

Sub Windows, vor fi citite următoarele fișiere:

Numele fișierului	Conținut
SYSTEMDIR`\my.ini`	Opțiuni globale
`C:\my.cnf`	Opțiuni specifice serverului
DATADIR`\my.cnf`	Opțiuni specifice utilizatorului

`DATADIR` reprezintă calea de acces spre catalogul de date din calculatorul dumneavoastră. (Aceasta este calea de acces compilată în server și nu poate fi modificată cu opțiunea `--datadir`.) Sub Windows, `DATADIR` este `C:\mysql\data`. `SYSTEMDIR` reprezintă calea de acces spre catalogul de sistem din Windows (de regulă ceva de genul `C:\Windows` sau `C:\WinNT`). Deși numele fișierelor cu opțiuni din Windows sunt specificate folosind `\` ca separator între componentele căii de acces, căile de acces specificate în cadrul fișierelor trebuie scrise folosind caracterul `/` drept separator. Dacă doriți să folosiți `\` ca separator, folosiți `\\` deoarece `\` este tratat drept caracter escape.

Fișierele cu opțiuni globale sunt folosite de către toate programele care recunosc fișierele cu opțiuni. Un fișier din catalogul de date al unui server este folosit numai de către serverul corespunzător. Fișierele specifice utilizatorilor sunt folosite de programe rulate de utilizatorul respectiv.

Opțiunile sunt specificate în grupuri. Iată un exemplu:

```
[client]
user=paul
password=secret

[mysql]
no-auto-rehash

[mysqlshow]
status
```

Numele grupurilor sunt date între paranteze drepte și corespund, de obicei, unui nume de program. În exemplul anterior, `[mysql]` indică grupul de opțiuni pentru clientul `mysql`, iar `[mysqlshow]` indică grupul de opțiuni pentru `mysqlshow`. Numele special de grup `[client]` vă permite să specificați opțiuni care se aplică tuturor programelor client. Programele client MySQL standard examinează atât grupul `[client]`, cât și grupul cu același nume ca al clientului. De exemplu, `mysql` examinează grupurile `[client]` și `[mysql]`, iar `mysqlshow` examinează grupurile `[client]` și `[mysqlshow]`.

Toate opțiunile care urmează după numele unui grup sunt asociate cu grupul respectiv. Un fișier cu opțiuni poate conține orice număr de grupuri, iar grupurile prezentate mai târziu au prioritate față de grupurile prezentate anterior. Dacă o opțiune dată apare de mai multe ori în grupurile pe care le examinează un program, se va folosi valoarea enumerată ultima.

Fiecare opțiune trebuie scrisă pe o linie separată. Primul cuvânt din linie este numele opțiunii, care trebuie specificat într-un format de tip nume lung, fără cratimele de început. (De exemplu, pentru a preciza opțiunea de comprimare în linia de comandă, puteți folosi `-C` sau `--compress`, dar într-un fișier cu opțiuni puteți folosi numai `compress`.) Orice opțiune în format lung acceptată de un program poate fi enumerată într-un fișier cu opțiuni. Dacă opțiunea necesită o valoare, specificați numele și valoarea separate printr-un caracter =.

Să examinăm următoarea linie de comandă:

```
mysql --compress --user=paul --set-variable max_allowed_packet=24M
```

Pentru a specifica aceleași informații într-un fișier cu opțiuni folosind grupul `[mysql]`, puteți proceda astfel:

```
[mysql]
compress
user=paul
set-variable=max_allowed_packet=24M
```

Observați că, într-un fișier cu opțiuni, `set-variable` este urmată de un caracter = în afara caracterului = dintre numele variabilei și valoarea acesteia.

Liniile vide ale fișierelor cu opțiuni sau liniile care încep cu unul din caracterele # sau ; sunt tratate drept comentarii și ignorate.

În valorile opțiunilor se pot folosi anumite secvențe escape pentru a indica anumite caractere speciale:

Secvență	Semnificație
`\b`	Ștergerea caracterului anterior cursorului
`\n`	Linie nouă
`\r`	Retur de car
`\s`	Spațiu
`\t`	Tabulator
`\\`	Backslash

Fiți atent să nu inserați în grupul `[client]` opțiuni care pot fi înțelese numai de către un singur client. De exemplu, dacă inserați în grupul `[client]` opțiunea `skip-line-numbers` specifică clientului `mysql`, veți descoperi subit că alte programe client, precum `mysqlimport`, nu mai funcționează. (Veți primi un mesaj de eroare, urmat de mesajul de asistență.) Mutați opțiunea `skip-line-numbers` în grupul `[mysql]` și totul va fi în regulă.

Opțiunile acceptate de către toate programele MySQL care citesc grupul `[client]` sunt `compress`, `debug`, `help`, `host`, `password`, `pipe`, `port`, `select`, `user` și `version`. (Nu are nici un sens să inserați `help` sau `version` într-un fișier cu opțiuni, desigur.)

Păstrați confidențialitatea fișierelor cu opțiuni specifice utilizatorului

Verificați ca fiecare fișier cu opțiuni specific utilizatorului să se afle în posesia utilizatorului adecvat și cu modul este configurat la 600 sau 400, astfel încât fișierul să nu poată fi citit de alți utilizatori. Este de dorit ca informațiile privind numele de utilizator și parola să nu fie accesibile nici unei alte persoane în afara utilizatorului pentru care este valabil fișierul.

Variabile de mediu

Programele MySQL examinează valorile a numeroase variabile de mediu pentru a obține valorile opțiunilor. Variabilele de mediu au o precedență redusă; opțiunile specificate folosind variabile de mediu pot fi anulate prin opțiuni specificate într-un fișier cu opțiuni sau în linia de comandă.

Programele MySQL verifică următoarele variabile de mediu.

- `MYSQL_DEBUG`

  Opțiunile de utilizat la depanare. Această variabilă nu are nici un efect dacă sistemul MySQL nu a fost construit cu suportul pentru depanare activat. Configurarea variabilei `MYSQL_DEBUG` este similară cu utilizarea opțiunii `--debug`.

- `MYSQL_PWD`

  Parola care se va folosi pentru stabilirea conexiunilor la serverul MySQL. Configurarea variabilei `MYSQL_PWD` este similară cu utilizarea opțiunii `--password`.

- `MYSQL_TCP_PORT`

  Pentru clienți, acesta este numărul de port care se va folosi la stabilirea unei conexiuni TCP/IP cu serverul. Pentru `mysqld`, acesta este portul la care serverul va aștepta conexiunile TCP/IP. Configurarea variabilei `MYSQL_TCP_PORT` este similară cu utilizarea opțiunii `--port`

- MYSQL_UNIX_PORT

  Pentru clienți, aceasta este calea de acces a fișierului soclu folosit la stabilirea conexiunilor cu socluri de domeniu UNIX la serverul care rulează pe localhost. Pentru mysqld, acesta este soclul la care se așteaptă conexiuni locale. Configurarea variabilei MYSQL_UNIX_PORT este similară cu folosirea opțiunii --socket.

- TMPDIR

  Calea de acces la catalogul în care se vor crea fișierele temporare. Configurarea acestei variabile este similară cu utilizarea opțiunii --tmpdir.

- USER

  Acesta este numele de utilizator MySQL care se va folosi pentru conectarea la server. Această variabilă este folosită numai de către programele client care rulează sub Windows; configurarea acesteia este similară cu utilizarea opțiunii --user.

  Clientul mysql verifică valoarea următoarelor variabile de mediu:

- MYSQL_HISTFILE

  Numele fișierului care va fi folosit pentru stocarea istoricului utilizării liniei de comandă. Valoarea prestabilită a acestei variabile, dacă nu este configurată, este $HOME/.mysql_history, unde $HOME este amplasamentul catalogului dumneavoastră de bază.

- MYSQL_HOST

  Gazda la care se va conecta clientul atunci când stabilește o conexiune cu serverul. Configurarea acestei variabile este similară cu utilizarea opțiunii --host.

isamchk și myisamchk

Aceste utilitare vă permit să verificați și să reparați tabelele deteriorate, să afișați informații despre tabele, să executați analiza distribuției valorilor cheie din index și să dezactivați, respectiv să activați indexurile. Capitolul 4, „Optimizarea interogărilor", conține mai multe informații cu privire la analiza cheilor și la dezactivarea indexurilor. Capitolul 13, „Întreținerea și repararea bazelor de date", furnizează mai multe informații despre verificarea și repararea tabelelor.

myisamchk este utilizat pentru tabele care folosesc formatul de stocare mai recent MyISAM. Aceste tabele prezintă nume de fișiere de date, respectiv index, cu sufixele .MYD și .MYI. Utilizați isamchk pentru tabele care folosesc mai vechiul format de stocare ISAM. Acestea folosesc nume de fișiere de date, respectiv index, cu sufixele .ISD și .ISM. Dacă indicați oricărui utilitar să opereze pe un tabel de un alt tip decât cel căruia îi este destinat, programul va afișa un mesaj de avertizare și va ignora tabelul.

Nu executați verificări sau reparări ale tabelelor în timp ce serverul MySQL are acces la tabel! Capitolul 13 discută despre metodele de a împiedica serverul să folosească un tabel în timp ce isamchk sau myisamchk lucrează cu tabelul respectiv.

Utilizare

```
isamchk [optiuni] nume_tabel[.ISM] ...
myisamchk [optiuni] nume_tabel[.MYI] ...
```

Dacă nu sunt specificate opțiuni, aceste utilitare verifică tabelele denumite, în căutarea erorilor. În caz contrar, tabelele sunt prelucrate în conformitate cu semnificația opțiunilor specificate. Un argument nume_tabel poate fi numele unui tabel sau numele fișierului index aferent tabelului (.ISM sau .MYI). Este convenabilă utilizarea numelor de fișiere index, deoarece puteți folosi caractere de înlocuire pentru nume de fișiere pentru ca programul să acționeze dintr-o singură comandă asupra tuturor tabelelor de un anumit tip de stocare. De exemplu, puteți verifica toate tabelele dintr-un catalog în acest mod:

```
% isamchk *.ISD
% myisamchk *.MYI
```

Aceste utilitare nu fac nici o presupunere cu privire la amplasamentul fișierelor tabel. Trebuie să specificați calea de acces spre fișierele pe care doriți să le folosiți, dacă acestea nu se află în catalogul curent. Deoarece se presupune că fișierele tabel nu sunt amplasate în catalogul de date al serverului, puteți copia fișierele tabel într-un alt catalog și puteți lucra folosind copiile în locul originalelor.

Unele dintre opțiuni se referă la numere de index. Indexurile sunt numerotate începând de la 1. Puteți emite o interogare SHOW INDEX sau puteți folosi comanda mysqlshow --keys pentru a descoperi numerotarea de index pentru un anumit tabel. Coloana Key_name afișează indexurile în aceeași ordine în care cele două programe „văd" indexurile respective.

MYSQL_PWD este un pericol la adresa securității

Utilizarea variabilei MYSQL_PWD pentru stocarea unei parole se constituie într-un risc de securitate, deoarece alți utilizatori ai sistemului dumneavoastră pot descoperi cu ușurință valoarea sa. De exemplu, utilitarul ps va afișa configurațiile variabilelor de mediu pentru alți utilizatori.

Opțiuni standard acceptate de isamchk și myisamchk

--debug	--set-variable	--verbose
--help	--silent	--version

Opțiunea --silent arată că vor fi afișate numai mesajele de eroare. Opțiunea --verbose afișează mai multe informații, atunci când este furnizată cu opțiunile --description sau --extend-check pentru oricare din cele două programe, respectiv când este dată cu opțiunea --check pentru myisamchk. Opțiunile --silent și --verbose pot fi specificate de mai multe ori, pentru a le spori efectul.

Opțiuni comune programelor isamchk și myisamchk

Cele două utilitare au numeroase opțiuni în comun:

- -a, --analyze

  Execută analiza distribuției cheilor. Această operație poate ajuta serverul să execute mai rapid căutări și uniri bazate pe index. Puteți obține informații despre distribuția

cheilor în urma analizei prin rularea din nou a utilitarelor `isamchk` sau `myisamchk`, cu opțiunile `--description` și `--verbose`.

- `-b n, --block-search=n`

  Afișează începutul rândului de tabel care conține un bloc, începând de la blocul *n*. Această opțiune se folosește numai la depanare.

- `-d, --description`

  Afișează informații semnificative referitoare la tabel.

- `-e, --extend-check`

  Execută o verificare amănunțită a tabelului. Utilizarea acestei opțiuni este rareori necesară, deoarece `myisamchk` și `isamchk` pot găsi toate erorile folosind unul din modurile de verificare mai puțin aprofundată.

- `-f, --force`

  Forțează verificarea sau repararea unui tabel chiar și atunci când există deja un fișier temporar pentru tabel. În mod normal, `myisamchk` și `isamchk` pur și simplu își încheie execuția după afișarea unui mesaj de eroare dacă găsesc un fișier *`nume_tabel`*`.TMD`, deoarece acest lucru poate însemna că o altă instanță a programului este deja în funcțiune. Totuși, fișierul mai poate exista și dacă ați suprimat o invocare anterioară a programului în timp ce acesta funcționa, caz în care fișierul poate fi eliminat fără riscuri. Dacă știți că aceasta este situația, folosiți opțiunea `--force` pentru indica oricăruia dintre utilitare să ruleze chiar dacă există un fișier temporar. (Alternativ, puteți elimina manual fișierul temporar.)

  Dacă folosiți opțiunea `--force` la verificarea tabelelor, programul va reporni automat cu opțiunea `--recover` pentru fiecare tabel unde au fost depistate probleme.

- `-i, --information`

  Afișează informații cu caracter statistic relativ la conținutul tabelului.

- `-k n, --keys-used=n`

  Se folosește cu opțiunea `--recover`. Indică programului MySQL să actualizeze numai primele *n* indexuri. Cu alte cuvinte, sunt dezactivate indexurile cu numere mai mari decât *n*. Această opțiune se poate folosi pentru îmbunătățirea performanțelor operațiilor `INSERT`, `DELETE` și `UPDATE`. Prin specificarea unui număr de index egal cu `0` se dezactivează toate indexurile. Prin specificarea unui număr de indexuri egal cu numărul de index cel mai mare se reactivează toate indexurile.

- `-l, --no-symlinks`

  Dacă un argument tabel este o legătură simbolică, în mod normal va fi reparat tabelul spre care indică legătura simbolică. Când este specificată această opțiune, legăturile simbolice sunt ignorate. În schimb, legătura simbolică este *înlocuită* cu versiunea nouă (reparată) a fișierului.

- `-q, --quick`

  Această opțiune este folosită în conjuncție cu `--recover` pentru o reparare mai rapidă decât atunci când se utilizează numai `--recover`. Fișierul de date este neatins atunci când sunt specificate ambele opțiuni. Pentru a forța programul să modifice fișierul de date chiar și atunci când sunt găsite valori duplicate ale cheilor, specificați opțiunea `--quick` de două ori. Dacă folosiți această opțiune de două ori, nu uitați să executați în prealabil o copie de siguranță a fișierului de date.

- `-r, --recover`

  Execută o operație normală de refacere. Aceasta poate remedia majoritatea problemelor, cu excepția valorilor duplicate aflate într-un index care ar trebui să fie unic.

- `-o, --safe-recover`

  Folosește o metodă de refacere mai lentă decât metoda `--recover`, dar care poate rezolva câteva probleme imposibil de remediat de către aceasta din urmă.

- `-S, --sort-index`

  Sortează blocurile de index, pentru accelerarea operațiilor de citire secvențială a blocurilor.

- `-R n, --sort-records=n`

  Sortează înregistrările de date conform ordinii în care sunt afișate înregistrările în indexul *n*. Operațiile ulterioare de regăsire bazate pe indexul dat vor avea o viteză mai mare. Prima dată când executați această operație asupra unui tabel, viteza de execuție poate fi foarte redusă, deoarece înregistrările dumneavoastră nu vor fi așezate în ordine.

- `-u, --unpack`

  Despachetează un fișier împachetat. `myisamchk` poate despacheta fișiere MyISAM împachetate cu `myisampack`; `isamchk` poate despacheta fișiere ISAM împachetate cu `pack_isam`. Această opțiune se poate folosi la conversia într-o formă modificabilă a unui tabel comprimat numai pentru citire.

- `-w, --wait`

  Dacă un tabel este blocat, așteaptă până când devine disponibil. Fără opțiunea `--wait`, programul va aștepta o blocare timp de 10 secunde și apoi va afișa un mesaj de eroare dacă nu poate obține nici o blocare.

Opțiuni specifice utilitarului `myisamchk`

Deși nu există opțiuni specifice programului `isamchk`, există numeroase opțiuni caracteristice utilitarului `myisamchk`.

- `-c, --check`

  Verifică tabelele în căutarea erorilor. Aceasta este acțiunea prestabilită dacă nu sunt specificate opțiuni.

- `-D n, --data-file-length=n`

  Specifică lungimea maximă permisă a fișierului de date atunci când se reconstruiește un fișier de date care s-a umplut. (Acest lucru se produce când un fișier atinge maxima dimensională impusă de MySQL sau de limitările impuse de sistemul dumneavoastră de operare asupra dimensiunilor fișierelor. De asemenea, se produce atunci când numărul de rânduri atinge limita impusă de structurile de date interne ale tabelelor.)

Valoarea este specificată în octeți. Această opțiune are efect numai atunci când este folosită cu `--recover` sau `--safe-recover`.

- `-F, --fast`

  Nu verifică tabelele care nu au suferit modificări de la ultima verificare.

- `-m, --medium-check`

  Verifică un tabel folosind o metodă mai rapidă decât metoda de verificare aprofundată, dar ceva mai puțin „pătrunzătoare" decât aceasta. (Mesajul de asistență al utilitarului `myisamchk` arată că această metodă găsește „numai" 99,99% din erori.) Acest mod de verificare este suficient în majoritatea situațiilor. Modul de verificare mediu lucrează prin calculul valorilor CRC ale cheilor din index și compararea acestora cu valorile CRC calculate din coloanele indexate din fișierul de date.

- `-T, --read-only`

  Nu marchează tabelul ca fiind verificat.

- `-A [n], --set-auto-increment[=n]`

  Forțează valorile `AUTO_INCREMENT` să înceapă de la *n* (respectiv de la o valoare mai mare, dacă tabelul conține deja înregistrări cu valorile `AUTO_INCREMENT` egale cu *n*). Dacă nu este specificată nici o valoare, această opțiune configurează următoarea valoare `AUTO_INCREMENT` astfel încât să fie cu o unitate mai mare decât valoarea curentă maximă.

- `-t nume_cale, --tmpdir=nume_cale`

  Precizează calea de acces a catalogului care se va utiliza pentru fișierele temporare. Valoarea prestabilită este valoarea variabilei de mediu sau `/tmp` dacă variabila respectivă nu este configurată.

Variabile pentru `isamchk` și `myisamchk`

Următoarele variabile folosite de utilitarele `isamchk` și `myisamchk` pot fi configurate cu opțiunea `--set-variable`:

- `key_buffer_size`

  Dimensiunea bufferului folosit pentru blocurile de index.

- `read_buffer_size`

  Dimensiunea bufferului de citire.

- `write_buffer_size`

  Dimensiunea bufferului de scriere.

- `sort_buffer_size`

  Dimensiunea bufferului folosit pentru operațiile de sortare după valorile cheilor. (Variabila este folosită pentru opțiunea `--recover`, nu pentru `--safe-recover`.)

- `sort_key_blocks`

  Această variabilă este corelată cu adâncimea structurii arborelui B folosit pentru index. Nu este necesar să o modificați.

- `decode_bits`

  Numărul de biți care se va folosi la decodarea tabelelor comprimate. Valori mai mar ale acestui parametru pot avea ca rezultat o operare mai rapidă, dar vor necesita o cantitate suplimentară de memorie. În general, valoarea prestabilită este suficientă.

`myisampack` și `pack_isam`

Utilitarele `myisampack` și `pack_isam` produc tabele comprimate, numai pentru citire Acestea realizează reduceri caracteristice ale spațiului de stocare necesar cuprinse între 40 și 70 de procente, păstrând însă o viteză mare de acces la înregistrări. Utilitaru `myisampack` împachetează tabelele MyISAM și funcționează cu toate tipurile de coloane Programul `pack_isam` împachetează tabelele ISAM și funcționează numai cu tabele care nu conțin coloane `BLOB` sau `TEXT`.

`myisampack` și `pack_isam` nu sunt incluse în MySQL în mod prestabilit, dar sunt disponibile pentru clienții care comandă minimum 10 licențe sau care beneficiază de un contract de asistență la nivelul de asistență extins sau la un nivel superior. Aceste prevederi se pot modifica astfel încât programele să fie disponibile în aceleași condiții ca și celelalte componente ale distribuției MySQL. (Vezi manualul de referință MySQL pentru informații curente privind asistența și licențele.)

Pentru citirea tabelelor care au fost împachetate cu aceste utilitare nu este necesară nici o versiune specială a sistemului MySQL, ceea ce face aceste programe utilizabile mai ales în aplicații pentru care dumneavoastră doriți să distribuiți un tabel conținând informații de arhivă sau de natură enciclopedică numai pentru citire și care nu trebuie actualizate. Dacă doriți să convertiți un fișier împachetat în forma despachetată și implicit modificabilă, puteți proceda astfel folosind `myisamchk --unpack` (pentru tabelele MyISAM), respectiv `isamchk --unpack` (pentru tabelele ISAM).

`myisampack` și `pack_isam` împachetează fișierele de date, dar nu se ating de fișierele index. Pentru a actualiza indexurile, rulați `myisamchk -rq` după ce ați rulat `myisampack` sau `isamchk -rq`, după ce ați rulat `pack_isam`.

Utilizare

```
myisampack [optiuni] nume_tabel ...
pack_isam [optiuni] nume_tabel ...
```

Un argument *nume_tabel* poate fi numele unui tabel sau numele fișierului index aferent tabelului (un fișier `.MYI` pentru tabelele MyISAM, respectiv un fișier `.ISM` pentru tabelele ISAM). Specificați calea de acces la catalogul în care este localizat tabelul, dacă nu vă aflați în catalogul respectiv.

Opțiuni standard acceptate de `myisampack` și `pack_isam`

`--debug`	`--silent`	`--version`
`--help`	`--verbose`	

Opțiuni comune programelor myisampack și pack_isam

- `-b, --backup`

  Indică utilitarelor myisampack sau pack_isam să execute o copie de siguranță pentru fiecare argument *nume_tabel* sub numele *nume_tabel*.OLD înainte de a-l împacheta.

- `-f, --force`

  Forțează împachetarea unui tabel chiar dacă fișierul împachetat rezultant este mai mare decât originalul sau dacă există deja un fișier temporar aferent tabelului. În mod normal, myisampack și pack_isam își încheie pur și simplu execuția după afișarea unui mesaj de eroare, dacă găsesc un fișier cu numele *nume_tabel*.TMD, deoarece acesta poate fi un indiciu că o altă instanță a programului este deja în funcțiune. Cu toate acestea, fișierul poate exista și dacă ați suprimat o invocare anterioară a programului în timpul rulării acestuia, caz în care fișierul poate fi eliminat fără probleme. Dacă știți că aceasta este situația, folosiți --force pentru a indica oricărui utilitar să împacheteze tabelul, chiar dacă fișierul temporar există.

- `-j unire_tbl, --join=unire_tbl`

  Unește (fuzionează) toate tabelele denumite în linia de comandă într-un singur tabel împachetat, denumit *unire_tbl*. Toate tabelele care vor fuziona trebuie să aibă aceeași structură. (Numele coloanelor, tipurile și indexurile trebuie să fie identice.)

- `-t, --test`

  Rulează în modul test. Este rulat un test de împachetare și se afișează informații despre rezultatele pe care le-ați obține dacă ați împacheta efectiv tabelul.

- `-T nume_cale, --tmpdir=nume_cale`

  Precizează numele căii de acces la catalogul care se va utiliza pentru fișierele temporare.

- `-W, --wait`

  Așteaptă și reia încercarea dacă există vreun tabel în uz. Nu doriți să împachetați un tabel dacă acesta poate fi actualizat în timp ce îl împachetați.

Opțiuni specifice utilitarului pack_isam

Deși nu există opțiuni specifice pentru myisampack, există una specifică utilitarului pack_isam, și anume -p *n*, --packlength=*n*. Această opțiune indică programului pack_isam să folosească un spațiu de stocare a lungimii înregistrării egal cu *n* octeți, unde *n* este un întreg cuprins între 1 și 3. Utilitarul pack_isam încearcă în mod automat să determine numărul octeților necesari pentru înregistrarea lungimii fiecărei înregistrări din tabelul împachetat. În unele situații, poate ajunge la concluzia că este posibilă utilizarea unei lungimi mai mici, caz în care pack_isam emite un mesaj în acest sens. Puteți rula pack_isam din nou și specificați în mod explicit lungimea mai mică folosind o opțiune --packlength pentru a obține unele economii suplimentare de spațiu.

mysql

Clientul mysql este un program interactiv care vă permite să vă conectați la server, s emiteți interogări și să vizualizați rezultatele. De asemenea, mysql mai poate fi folosit în modul batch, pentru a executa interogări stocate într-un fișier, dacă redirecționat intrarea comenzii astfel încât să citească din fișierul respectiv. De exemplu:

```
% mysql -u paul -p -h pit-viper.snake.net samp_db < my_query_file
```

În modul interactiv, când mysql pornește, afișează un prompt mysql pentru a arăta c așteaptă datele de intrare. Pentru a emite o interogare, tastați-o (folosind mai multe linii dacă este necesar) și apoi indicați sfârșitul interogării tastând ; (caracterul punct și vir gulă) sau \g. mysql trimite interogarea la server, afișează rezultatele și apoi afișează u alt prompt, indicând că este pregătit pentru o altă interogare.

În mysql, promptul poate lua diferite forme, pentru a indica datele de intrare pe care l așteaptă:

mysql	Așteaptă prima linie a unei noi interogări
-	Așteaptă linia următoare a interogării curente
'	Așteaptă finalizarea unui șir cu ghilimele simple din interogarea curentă
"	Așteaptă finalizarea unui șir cu ghilimele duble din interogarea curent

Prompturile ' și " arată că ați început să scrieți un șir cu ghilimele simple, respectiv dubl într-o linie anterioară și nu ați introdus încă ghilimelele de închidere. De regulă, aceast se întâmplă când ați uitat să încheiați un șir. Dacă aceasta este situația, pentru a părăs modul de colectare a șirurilor, introduceți ghilimelele de închidere adecvate indicate d prompt, urmate de expresia \c pentru a anula interogarea curentă.

Când programul mysql este folosit în modul interactiv, salvează interogările într-u fișier istoric. Acest fișier este $HOME/.mysql_history în mod prestabilit sau poate fi spe cificat în mod explicit prin configurarea variabilei de mediu MYSQL_HISTORY. Interogăril pot fi reluate din istoricul comenzilor și emise din nou, cu sau fără alte modificări. List următoare prezintă unele dintre aceste comenzi de editare (este posibil ca tastele c săgeți să nu funcționeze sub Windows):

Secvență de taste	Semnificație
Săgeată sus, Ctrl-P	Reia linia anterioară
Săgeată jos, Ctrl-N	Reia linia următoare
Săgeată stânga, Ctrl-B	Mută cursorul la stânga (înapoi)
Săgeată dreapta, Ctrl-F	Mută cursorul la dreapta (înainte)
Escape Ctrl-B	Mută cursorul înapoi cu un cuvânt
Escape Ctrl-F	Mută cursorul înainte cu un cuvânt
Ctrl-A	Mută cursorul la începutul rândului
Ctrl-E	Mută cursorul la sfârșitul rândului
Ctrl-D	Șterge caracterul plasat sub cursor

Secvență de taste	Semnificație
Delete	Șterge caracterul plasat la stânga cursorului
Escape D	Șterge cuvântul
Escape Backspace	Șterge cuvântul plasat la stânga cursorului
Ctrl-K	Șterge toate caracterele de la cursor la sfârșitul liniei
Ctrl-_	Anulează ultima modificare; poate fi repetat

Unele opțiuni nu permit utilizarea fișierului istoric. În general, acestea sunt opțiuni care indică utilizarea non-interactivă a programului `mysql`, precum `--batch`, `--html` și `--quick`.

Utilizare

```
mysql [optiuni] [nume_db]
```

Dacă specificați un argument *nume_db*, baza de date respectivă devine baza de date curentă (prestabilită) pentru sesiunea dumneavoastră. Dacă nu specificați un asemenea argument, `mysql` începe fără nici o bază de date curentă și va trebui fie să completați toate referințele la tabele cu numele unei baze de date, fie să emiteți o instrucțiune `USE` *nume_db* pentru a specifica o bază de date prestabilită.

Opțiuni standard acceptate de `mysql`

`--compress`	`--password`	`--silent`	`--version`
`--debug`	`--pipe`	`--socket`	
`--help`	`--port`	`--user`	
`--host`	`--set-variable`	`--verbose`	

Opțiunile `--silent` și `--verbose` pot fi specificate de mai multe ori, pentru a se obține un efect sporit al acestora.

Opțiuni specifice programului `mysql`

- `-B, --batch`

  Precizează că `mysql` trebuie să ruleze în modul batch. Rezultatele interogărilor sunt afișate într-un format delimitat prin tabulatori (fiecare rând pe o linie separată, cu tabulatori între valorile din coloane). Acest lucru este convenabil în special pentru generarea datelor de ieșire pe care doriți să le importați într-un alt program, precum o foaie de calcul tabelar. Rezultatele interogărilor includ, în mod prestabilit, un rând inițial de antete de coloană. Pentru a elimina aceste antete, folosiți opțiunea `skip-column-names`.

- `-T, --debug-info`

  Afișează informații de depanare atunci când programul își încheie execuția.

- `-e` *`interogare`*`, --execute=`*`interogare`*

  Execută interogarea și își încheie execuția. Trebuie să includeți interogarea între ghilimele, pentru a preveni interpretarea acesteia de către shell sub formă de multiple argumente în linia de comandă. Se pot specifica mai multe interogări, separându-le prin caractere punct și virgulă în șirul `interogare`.

- `-f, --force`

  În mod normal, când `mysql` citește interogări dintr-un fișier, își încheie execuția dacă se produce vreo eroare. Această opțiune determină programul `mysql` să continue prelucrarea interogărilor, indiferent de erori.

- `-H, --html`

  Generează date de ieșire în format HTML. Opțiunea a devenit funcțională în MySQL 3.22.26.

- `-i, --ignore-space`

  Indică serverului să ignore spațiile dintre numele funcțiilor și caracterul paranteză rotundă de deschidere care inițiază lista cu argumente. În mod normal, numele funcțiilor trebuie să fie urmate imediat de către paranteze, fără spații intermediare.

- `-A --no-auto-rehash`

  Când `mysql` pornește, calculează valorile hash ale numelor bazelor de date, tabelelor și coloanelor pentru a construi o structură de date care permite completarea rapidă a numelor. (Puteți tasta partea inițială a unui nume atunci când introduceți o interogare și apoi apăsați pe tasta Tab; `mysql` va completa numele, dacă nu există ambiguități.) Această opțiune elimină calculul valorilor hash, ceea ce permite programului `mysql` să pornească mai rapid, în special dacă dispuneți de multe tabele. Dacă doriți să folosiți caracteristica de completare a numelor după pornirea programului `mysql`, puteți folosi comanda `rehash` la promptul `mysql`.

- `-o --one-database`

  Această opțiune este folosită când actualizați baze de date pornind de la conținutul unui fișier jurnal de actualizare. Opțiunea indică programului MySQL să actualizeze numai baza de date prestabilită (baza de date denumită în linia de comandă). Actualizările aduse altor baze de date sunt ignorate. Dacă nici o bază de date nu este denumită în linia de comandă, nu sunt executate actualizări.

- `-q, --quick`

  În mod normal, `mysql` regăsește integral rezultatul unei interogări de la server înainte de a-l afișa. Această opțiune determină afișarea fiecărui rând pe măsură ce rândul respectiv este regăsit, utilizându-se astfel o cantitate mult mai redusă de memorie și permițând executarea cu succes a unor interogări de mari dimensiuni, care în alte condiții ar fi eșuat. Totuși, această opțiune nu trebuie specificată pentru uz interactiv; dacă utilizatorul oprește temporar datele de ieșire sau suspendă programul `mysql`, serverul continuă să aștepte, ceea ce poate afecta funcționarea altor clienți.

- `-r, --raw`

  Scrie valorile coloanelor fără a modifica semnificația eventualelor caractere speciale. Această opțiune este folosită în conjuncție cu opțiunea `--batch`.

- `-N, --skip-column-names`

  Inhibă afișarea numelor de coloane ca antete de coloană în rezultatele interogărilor. Puteți obține acest efect și prin specificarea de două ori a opțiunii `--silent`.

- `-L, --skip-line-numbers`

Nu scrie numerele liniilor atunci când se produc erori.

- `-t, --table`

  Produce datele de ieșire în format tabular, cu valorile din fiecare rând delimitate de bare și aliniate pe verticală.

- `-n, --unbuffered`

  Golește după fiecare interogare bufferul folosit pentru comunicații cu serverul.

- `-E, --vertical`

  Afișează rezultatele interogării pe verticală, adică fiecare rând al unui set de rezultate este afișat sub formă de set de linii, câte o linie pentru fiecare coloană. Fiecare linie constă din numele și valoarea unei coloane. Afișarea fiecărui rând este precedată de o linie care indică numărul rândului din cadrul setului de rezultate. Formatul de afișare pe verticală poate fi util atunci când o interogare generează linii foarte lungi.

  Dacă această opțiune nu este specificată, puteți activa formatul de afișare pe verticală pentru interogările individuale prin utilizarea caracterului de terminare `\G`, în loc de `;` sau `\g`.

  Această opțiune a fost introdusă în versiunea MySQL 3.22.5.

- `-w, --wait`

  Dacă o conexiune cu serverul nu poate fi stabilită, așteaptă și reia încercarea.

Variabile pentru `mysql`

Următoarele variabile pentru `mysql` pot fi configurate cu opțiunea `--set-variable`:

- `max_allowed_packet`

  Dimensiunea maximă a bufferului folosit pentru comunicația între server și client.

- `net_buffer_length`

  Dimensiunea inițială a bufferului folosit pentru comunicația între server și client. Acest buffer poate fi extins până la maximum `max_allowed_packet` octeți.

Comenzi folosite în `mysql`

Programul `mysql` înțelege numeroase comenzi și vă permite să expediați instrucțiuni SQL serverului MySQL. Fiecare comandă trebuie emisă pe o singură linie; la sfârșitul liniei nu este necesar caracterul punct și virgulă. Majoritatea comenzilor au o formă lungă, alcătuită dintr-un cuvânt, respectiv o formă scurtă, care constă dintr-un caracter backslash urmat de o singură literă. Comenzile în forma lungă nu sunt sensibile la diferența între majuscule și minuscule. Comenzile în formă scurtă trebuie specificate folosind dimensiunile literelor prezentate în lista următoare:

- `clear, \c`

  Șterge (anulează) interogarea curentă. Interogarea curentă este acea interogare pe care o tastați la un moment dat; această comandă nu anulează o interogare care a fost deja trimisă serverului și pentru care `mysql` afișează date de ieșire.

- `connect [nume_db [nume_gazda]], \r [nume_db [nume_gazda]]`

  Se conectează la baza de date specificată sau la gazda menționată. Dacă numele bazei de date sau numele gazdei lipsesc, se vor utiliza valorile cel mai recent folosite din sesiunea `mysql` curentă.

- `edit, \e`

  Editează interogarea curentă. Editorul care se va utiliza este determinat de valoarea variabilelor de mediu `EDITOR` și `VISUAL`, respectiv este `vi` dacă nici una dintre acestea nu este configurată.

- `ego, \G`

  Trimite interogarea curentă la server și afișează rezultatul pe verticală. Această comandă a fost introdusă în versiunea MySQL 3.22.11.

- `exit`

  Similară cu `quit`.

- `go, \g, ;`

  Trimite interogarea curentă la server și afișează rezultatul.

- `help, \h, ?`

  Afișează un mesaj de asistență, care descrie comenzile `mysql` disponibile.

- `print, \p`

  Afișează interogarea curentă (textul interogării în sine, nu rezultatele obținute prin executarea interogării).

- `quit, \q`

  Încheie execuția programului `mysql`.

- `rehash, \#`

  Recalculează informațiile necesare pentru completarea numelor bazelor de date, ale tabelelor și ale coloanelor. Vezi descrierea opțiunii `--no-auto-rehash`.

- `status, \s`

  Regăsește și afișează informațiile de stare de la server. Este o opțiune utilă dacă doriți să verificați versiunea serverului și baza de date curentă.

- `use nume_db, \u nume_db`

  Selectează baza de date specificată pentru a o transforma în bază de date curentă.

mysql.server

mysql.server este un script care pornește și oprește serverul mysqld invocând scriptul safe_mysqld.

Utilizare

```
mysql.server start
mysql.server stop
```

În mod normal, mysql.server este folosit pe un sistem stil System V și este instalat într-unul din cataloagele de rulare din catalogul /etc. Sistemul pornește serverul invocând scriptul cu argumentul start în momentul încărcării sistemului de operare. Sistemul închide serverul invocând scriptul cu argumentul stop în momentul opririi sistemului. De asemenea, scriptul poate fi invocat manual, cu argumentul adecvat, pentru a porni, respectiv opri serverul.

mysqlaccess

Acest script vă permite să vă conectați la un server, să regăsiți informații despre privilegiile de acces și să testați rezultatele specificării privilegiilor de utilizator. Scriptul execută aceste operații folosind copii ale tabelelor user, db și host din baza de date mysql. (mysqlaccess nu se poate folosi pentru exersarea utilizării privilegiilor la nivel sau de coloană.) De asemenea, puteți înregistra schimbările din tabelele temporare în tabelele efective din baza de date mysql.

Pentru a folosi mysqlaccess, trebuie să aveți suficiente privilegii pentru a obține personal accesul la tabelele de acordare.

Utilizare

```
mysqlaccess [nume_gazda [nume_utilizator [nume_db]]] optiuni
```

Opțiuni standard acceptate de mysqlaccess

--host --password --user --version

Opțiuni specifice programului mysqlaccess

- -b, --brief

  Afișează rezultatele în format cu o singură linie.
- --commit

  Copiază tabelele de acordare temporare înapoi în baza de date mysql. Nu uitați să executați ulterior o comandă mysqladmin flush-privileges, astfel încât serverul să remarce schimbările.
- --copy

  Încarcă tabelele de acordare în tabelele temporare.

- -d *nume_db*, --db=*nume_db*

  Specifică numele bazei de date.
- --debug=*n*

  Specifică nivelul de depanare. *n* trebuie să fie un întreg cuprins între 0 și 3.
- --howto

  Afișează unele exemple privind modul de utilizare a programului mysqlaccess.
- --old-server

  Arată că serverul este mai vechi decât versiunea MySQL 3.21. Aceasta determină programul mysqlaccess să aducă anumite modificări interogărilor pe care le trimite la server.
- --plan

  Afișează o listă cu îmbunătățirile planificate pentru versiunile viitoare ale programului mysqlaccess.
- --preview

  Afișează diferențele în materie de privilegii între tabelele de acordare temporare și efective.
- --relnotes

  Prezintă informațiile de lansare ale programului mysqlaccess.
- -H *nume_gazda*, --rhost=*nume_gazda*

  Specifică gazda situată la distanță a serverului la care se va stabili conexiunea.
- --rollback

  Anulează modificările aduse tabelelor de acordare temporare.
- -P *parola*, --spassword=*parola*

  Specifică parola pentru super-utilizatorul MySQL (un utilizator cu privilegii suficiente pentru modificarea tabelelor de acordare).
- -U *nume_utilizator*, --superuser=*nume_utilizator*

  Specifică numele de utilizator al super-utilizatorului MySQL.
- -t, --table

  Afișează rezultatele în format cu tabulatori.

mysqladmin

Utilitarul mysqladmin comunică cu serverul MySQL pentru a executa o varietate de operații administrative. Puteți folosi mysqladmin pentru a obține informații de la server sau pentru a controla operarea serverului, pentru a configura parole și pentru a crea sau șterge bazele de date.

Utilizare

```
mysqladmin [optiuni] comanda ...
```

Opțiuni standard acceptate de mysqladmin

--compress	--host	--port	--user
--debug	--password	--silent	--version
--help	--pipe	--socket	

Opțiunea -- silent determină programul mysqladmin să-și încheie execuția în liniște, dacă nu se poate conecta la server.

Opțiuni specifice programului mysqladmin

- -f, --force

  Această opțiune are două efecte. Mai întâi, determină programul mysqladmin să nu ceară confirmare pentru comanda drop *nume_db*. În al doilea rând, când sunt specificate mai multe comenzi în linia de comandă, mysqladmin încearcă să execute fiecare comandă, chiar dacă se produc erori. În mod normal, programul mysqladmin își va încheia execuția după prima eroare.

- --relative

  Afișează diferența dintre valoarea curentă și valoarea anterioară atunci când se folosește cu --sleep. În prezent, această opțiune funcționează numai alături de comanda extended-status.

- -i *n*, --sleep=*n*

  Execută în mod repetat comenzile indicate în linia de comandă, cu un interval de *n* secunde între fiecare repetiție.

- -t *n*, --timeout=*n*

  Așteaptă *n* secunde înainte de a expira atunci când încearcă să se conecteze la server. Această opțiune a fost introdusă în MySQL versiunea 3.22.1.

- -w [*n*], --wait[=*n*]

  Specifică de câte ori programul trebuie să aștepte și să încerce din nou dacă nu a reușit să stabilească o conexiune cu serverul. Valoarea prestabilită a lui *n* este 1, dacă nu este specificată nici o valoare.

Comenzile din programul mysqladmin

După opțiunile din linia de comandă, puteți specifica una sau mai multe din următoarele comenzi. Numele comenzii poate fi redus la un prefix, cu condiția ca prefixul să fie unic. De exemplu, comanda processlist poate fi redusă la process sau proc, dar nu la forma p.

Multe din aceste comenzi au un echivalent SQL. Acestea sunt notate în cadrul descrierilor. Pentru mai multe informații cu privire la semnificația instrucțiunilor SQL, vezi Anexa D, „Referință de sintaxă SQL".

- create nume_db

  Creează o bază de date nouă, cu numele dat.

  Această comandă este similară comenzii SQL CREATE DATABASE *nume_db*.

- drop *nume_db*

  Șterge baza de date cu numele dat, precum și toate tabelele care se găsesc în baza de date. Fiți atent cu această comandă; nu vă puteți recupera baza de date după ce ați șters-o. Dacă nu a fost menționată opțiunea --force, programul mysqladmin cere confirmarea pentru execuția comenzii.

  Această comandă este similară comenzii SQL DROP DATABASE *nume_db*.

- extended-status

  Afișează numele și valorile variabilelor de stare ale serverului.

  Această comandă este similară cu instrucțiunea SQL SHOW STATUS.

  Comanda extended-status a fost introdusă în MySQL 3.22.10.

- flush-hosts

  Golește zona cache aferentă gazdelor.

  Această comandă este similară instrucțiunii SQL FLUSH HOSTS.

- flush-logs

  Golește (închide și redeschide) fișierele jurnal.

  Această comandă este similară instrucțiunii SQL FLUSH LOGS.

- flush-privileges

  Reîncarcă tabelele de acordare.

  Această comandă este similară instrucțiunii SQL FLUSH PRIVILEGES.

- flush-status

  Șterge variabilele de stare. (Astfel, numeroase contoare sunt reinițializate la valoarea zero.)

  Această comandă este similară instrucțiunii SQL FLUSH STATUS.

- flush-tables

  Golește zona cache aferentă tabelelor.

  Această comandă este similară instrucțiunii SQL FLUSH TABLES.

- kill *id,id* ...

  Suprimă firele de server specificate de numerele de identificator date. Dacă specificați mai multe numere, lista de identificatori nu trebuie să conțină spații, astfel încât să nu fie confundată cu alte comenzi după comanda kill. Pentru a identifica firele ce rulează la un moment dat, folosiți mysqladmin processlist.

  Această comandă este similară unei instrucțiuni SQL KILL pentru fiecare identificator de fir.

- password *parola_noua*

  Modifică parola aferentă utilizatorului MySQL sub al cărui cont v-ați conectat atunci când rulați comanda mysqladmin. Noua parolă este *parola_noua*. Trebuie să vă puteți conecta la server sub contul utilizatorului MySQL a cărui parolă este modificată, ceea ce servește ca o confirmare a faptului că știți vechea parolă a utilizatorului.

  Această comandă este similară instrucțiunii SQL SET PASSWORD.

  Comanda PASSWORD a fost introdusă în MySQL 3.22.
- ping

  Verifică dacă serverul MySQL rulează sau nu.
- processlist

  Afișează lista firelor de server care se execută la un moment dat.

  Această comandă este similară instrucțiunii SQL SHOW PROCESSLIST.
- refresh

  Golește zona cache a tabelului și închide și redeschide fișierele jurnal.
- reload

  Reîncarcă tabelele de acordare.

  Această comandă este similară instrucțiunii SQL FLUSH PRIVILEGES.
- shutdown

  Indică serverului să-și înceteze execuția.
- status

  Afișează un scurt mesaj de stare de la server.
- variables

  Afișează numele și valorile variabilelor serverului.

  Această comandă este similară instrucțiunii SQL SHOW VARIABLES.
- version

  Regăsește și afișează șirul cu informații privind versiunea serverului.

mysqlbug

Capitolul 2 al manualului de referință MySQL conține o procedură detaliată pentru completarea unui raport de semnalare a unei hibe. Prin respectarea acestei proceduri, vă asigurați că raportul dumneavoastră va fi util și va conține suficiente informații pentru rezolvarea problemei pe care ați semnalat-o. O parte cheie a acestei proceduri constă în utilizarea scriptului mysqlbug. Acesta este folosit pentru crearea și expedierea unui raport de semnalare a unei hibe către lista de corespondență MySQL atunci când descoperiți o problemă a sistemului MySQL. Programul mysqlbug adună informații despre sistemul dumneavoastră și despre configurația MySQL și apoi vă deschide un editor care include conținutul mesajului poștal care va fi trimis. Editați mesajul pentru a adăuga cât mai multe informații cu putință despre problema pe care o semnalați și apoi scrieți mesajul pentru a vă salva modificările și părăsiți editorul. mysqlbug vă va întreba dacă doriți sau nu să trimiteți raportul, apoi îl trimite, dacă dumneavoastră confirmați intenția de a-l trimite.

Vă rugăm să folosiți mysqlbug pentru a semnala hibele, dar nu-l utilizați oricum. În multe situații, o „hibă" nu este deloc... o hibă sau este un raport privind un fapt care a fost deja documentat în manualul de referință. De exemplu, de multe ori s-a semnalat ca hibă faptul că instrucțiunea DELETE FROM *nume_tabel* raportează un număr al rândurilor afectate egal cu zero. Este adevărat, dar acesta este comportamentul scontat. De asemenea, este documentat în manual ca atare, ca și interogarea pe care trebuie să o folosiți dacă doriți numărul real de rânduri afectate. Arhivele listei de corespondență MySQL reprezintă o altă sursă utilă de informații. Legăturile spre arhive se pot găsi la adresa http://www.mysql.com/doc.html.

Utilizare

```
mysqlbug [adresa]
```

Raportul de semnalare a unei hibe este trimis către lista de corespondență MySQL în mod prestabilit. Dacă specificați în linia de comandă o adresă de e-mail, raportul va fi trimis la adresa respectivă. Puteți specifica propria dumneavoastră adresă pentru a testa programul mysqlbug fără a trimite raportul spre lista de corespondență; este chiar recomandat să procedați astfel atunci când folosiți scriptul pentru prima dată și poate că nu sunteți sigur cu privire la modul de funcționare al acestuia.

mysqld

mysqld este serverul MySQL. Furnizează accesul programelor client la bazele de date, deci este necesar ca programul să funcționeze, în caz contrar clienții neputând folosi bazele de date administrate de server. Când mysqld pornește, deschide porturile de rețea la care va asculta și apoi așteaptă conexiunile client. Programul mysqld este multifir și prelucrează fiecare conexiune client folosind un fir separat pentru a asigura concurența între clienți. Interogările care scriu în baza de date sunt executate separat; când serverul începe să execute o asemenea interogare, nu va mai executa nici o altă interogare pentru tabelul implicat decât atunci când interogarea curentă și-a încheiat execuția. Din acest motiv, niciodată doi clienți nu vor putea modifica simultan același rând dintr-un tabel.

Utilizare

```
mysqld [optiuni]
```

Opțiuni standard acceptate de mysqld

--debug	--port	--user
--help	--socket	--version

Rețineți că, deși opțiunea --socket este acceptată, opțiunea -S nu este, deoarece serverul folosește opțiunea -Sg cu o semnificație care nu are nici o legătură cu fișierul soclu.

Dacă este dată opțiunea --user, aceasta specifică numele de utilizator pentru contul care se va folosi la rularea serverului. În acest caz, când serverul pornește, caută numele de utilizator și identificatorul de grup ale contului din fișierul parolă și apoi modifică

numele de utilizator și identificatorul de grup proprii astfel încât să fie similare celor ale contului. În acest mod, serverul rulează cu privilegiile asociate utilizatorului respectiv, nu cu privilegiile utilizatorului `root`. (Serverul trebuie să fie pornit sub contul `root` pentru ca opțiunea `--user` să fie eficientă; în caz contrar, nu va putea să-și modifice identificatorul de utilizator.) Această opțiune a fost introdusă în MySQL versiunea 3.22.

Opțiuni specifice serverului `mysqld`

- `-b` *`nume_cale`*, `--basedir=`*`nume_cale`*

  Specifică numele căii de acces spre catalogul de instalare MySQL. Multe alte căi de acces sunt rezolvate începând de la acest catalog, dacă sunt specificate sub formă de căi relative de acces.

- `--big-tables`

  Permite prelucrarea unor seturi de rezultate de mari dimensiuni prin salvarea tuturor rezultatelor temporare pe disc, nu prin stocarea lor în memorie. Astfel, sunt evitate majoritatea mesajelor de tip „tabel plin" care se produc ca rezultat al existenței unei cantități prea mici de memorie pentru stocarea unor seturi de rezultate de mari dimensiuni. Începând cu versiunea MySQL 3.23, această opțiune nu mai este necesară.

- `--bind-address=`*`adresa_ip`*

  Se asociază la adresa IP dată. În mod normal, `mysqld` se asociază la adresa IP prestabilită a gazdei la care rulează serverul. Această opțiune se poate folosi pentru a selecta o adresă alternativă de asociere, dacă o gazdă are mai multe adrese.

- `--bootstrap`

  Această opțiune este folosită de scripturile de instalare la prima instalare a sistemului MySQL.

- `--chroot=`*`nume_cale`*

  Rulează serverul MySQL ancorat la un anumit catalog rădăcină. Opțiunea precizează calea de acces spre catalogul respectiv. Pentru mai multe informații privind rularea într-un mediu `chroot()`, consultați pagina de manual UNIX aferentă comenzii `chroot()`. Această opțiune a fost introdusă în versiunea MySQL 3.22.2.

- `-h` *`nume_cale`*, `--datadir=`*`nume_cale`*

  Specifică numele căii de acces spre catalogul de date MySQL.

- `--default-table-type=`*`tip`*

  Specifică tipul prestabilit de stocare a tabelelor. Valoarea *`tip`* trebuie să fie `isam`, `myisam` sau `heap`. (Valoarea nu este sensibilă la diferența între majuscule și minuscule.) În mod prestabilit, dacă această opțiune nu este specificată, se folosesc tabelele MyISAM. Această opțiune a fost introdusă în versiunea MySQL 3.23.

- `--enable-locking`

  Activează blocarea externă (blocarea sistemului de fișiere) pentru sisteme precum Linux, unde opțiunea prestabilită este `--skip-locking`. Această opțiune a fost introdusă în MySQL 3.22.4.

- `-T [`*`n`*`], --exit-info[=`*`n`*`]`

  Afișează informațiile de depanare la încheierea execuției programului. Această opțiune a fost introdusă în versiunea MySQL 3.22.

- `--init-file=`*`nume_fisier`*

  Specifică un fișier cu instrucțiuni SQL care vor fi executate la pornire.

- `-L, --language=`*`nume_limba`*

  Afișează clienților mesaje de eroare în limba specificată. În mod normal, *`nume_limba`* va fi o valoare de genul `english` sau `german`.

- `-l [`*`nume_cale`*`], --log[=`*`nume_cale`*`]`

  Activează caracteristica de consemnare în jurnalul general. Acesta conține informații privind conexiunile clienților și interogările. Dacă parametrul *`nume_cale`* nu este dat, numele fișierului jurnal este *`nume_gazda`*`.log`, unde *`nume_gazda`* este numele gazdei serverului. Dacă *`nume_gazda`* este dat sub formă de cale relativă, aceasta va fi interpretată pornind de la catalogul de date.

- `--log-isam[=`*`nume_fisier`*`]`

  Activează caracteristica de consemnare în fișierele index. Această comandă se folosește numai pentru depanarea tabelelor ISAM/MyISAM.

- `--log-long-format`

  Scrie informații suplimentare în jurnalul de actualizare. Această opțiune a fost introdusă în versiunea MySQL 3.22.7.

- `--log-update[=`*`nume_cale`*`]`

  Activează consemnarea în jurnalul de actualizare. Jurnalul de actualizare conține textul oricărei interogări care modifică tabelele dintr-o bază de date. Dacă nu este dat parametrul *`nume_cale`*, numele fișierului jurnal de actualizare este *`nume_gazda`*`.`*`nnn`*, unde *`nume_gazda`* este numele gazdei serverului, iar *`nnn`* este un număr dintr-o secvență cu o unitate mai mare (numărul, nu secvența – N.T.) decât cel al jurnalului de actualizare anterior.

  Dacă *`nume_cale`* este dat și componenta finală nu conține extensii, serverul adaugă o extensie numerică de forma *`nnn`*, având o valoare conform celor specificate anterior. În cazul în care componenta finală a numelui de cale include o extensie, numele respectiv va fi folosit, fără modificări, ca nume de fișier pentru jurnalul de actualizare. Dacă *`nume_cale`* este specificat sub formă de cale relativă, este interpretat pornind de la catalogul de date.

- `--low-priority-updates`

  Acordă operațiilor de scriere o prioritate mai redusă decât operațiilor de regăsire. Această opțiune a fost introdusă în versiunea MySQL 3.23. (De la versiunea MySQL 3.22.5 și până la versiunea 3.23, opțiunea a purtat numele `--low-priority-inserts`.)

- `-n --new`

  Folosește rutine noi, posibil nesigure. Acestea sunt caracteristici ale sistemului MySQL care nu au fost încă declarate stabile. Utilizați această opțiune numai dacă aveți spirit de aventură.

- `-o, --old-protocol`

  Folosește protocolul care a fost utilizat pentru comunicațiile client-server anterior versiunii MySQL 3.21. Această opțiune poate fi necesară dacă serverul comunică cu programe client foarte vechi.

- `--one-thread`

  Rulează folosind un singur fir; folosit pentru depanare sub Linux, care în mod normal folosește minimum trei fire. Această opțiune a fost introdusă în versiunea MySQL 3.22.2.

- `--pid-file=`*`nume_cale`*

  Când programul `mysqld` pornește, își scrie identificatorul de proces (PID) într-un fișier. Această opțiune specifică numele căii de acces a fișierului PID. Fișierul poate fi folosit de alte procese pentru a determina numărul de proces al serverului, în mod caracteristic pentru trimiterea unui semnal. De exemplu, `mysql.server` citește fișierul atunci când trimite serverului un semnal de închidere.

- `--safe-mode`

  Această opțiune este similară opțiunii `--skip-new`, dar dezactivează un număr mai mare de facilități. O puteți încerca dacă MySQL pare instabil sau dacă anumite interogări complexe tind să determine rezultate incorecte. Dacă utilizarea acestei opțiuni îmbunătățește funcționarea serverului, vă rugăm să notați acest fapt atunci când folosiți programul `mysqlbug` pentru a raporta problemele pe care le întâlniți.

- `-Sg, --skip-grant-tables`

  Nu folosește tabelele de acordare pentru a verifica conexiunile clienților. Astfel, toți clienții primesc drepturi de acces total, pentru a executa orice operații. Puteți cere serverului să înceapă a folosi tabelele de acordare emițând o comandă `mysqladmin flush-privileges`.

- `--skip-host-cache`

  Dezactivează utilizarea zonei cache cu numele gazdei.

- `--skip-locking`

  Nu permite utilizarea blocării de sistem (blocarea sistemului de fișiere). Dacă folosiți această opțiune, trebuie să opriți serverul înainte de a folosi `isamchk` sau `myisamchk` pentru a repara tabelele; în caz contrar, riscați să le deteriorați.

- `--skip-name-resolve`

  Dacă este specificată această opțiune, tabelele de acordare trebuie să specifice gazdele în funcție de numărul IP sau folosind expresia `localhost`.

- `--skip-networking`

  Nu permite stabilirea conexiunilor TCP/IP. Clienții trebuie să se conecteze de la gazda locală, folosind `localhost`.

- `--skip-new`

  Nu folosește rutinele noi, posibil nesigure. Vezi opțiunea `--new`.

- `--skip-show-database`

  Nu permite utilizatorilor fără privilegii să emită interogări `SHOW DATABASES` sau să folosească instrucțiunea `SHOW TABLES` în baze de date pentru care nu au acces. Această opțiune a fost introdusă în MySQL versiunea 3.23.

- `--skip-thread-priority`

  În mod normal, interogările care modifică tabele rulează la o prioritate mai ridicată decât cele care regăsesc date. Dacă nu se dorește acest lucru, această opțiune determină serverul să nu acorde priorități diferite pentru diverse tipuri de interogări.

- `-t` *`nume_cale`*`, --tmpdir=`*`nume_cale`*

  Specifică numele căii de acces a catalogului care se va folosi pentru fișierele temporare. Această opțiune a fost introdusă în versiunea MySQL 3.22.4.

Opțiunile specifice sistemului Windows sunt următoarele:

- `-console`

  Afișează o fereastră a consolei pentru mesaje de eroare. Această opțiune a fost introdusă în versiunea MySQL 3.22.4.

- `--flush`

  Evacuează toate tabelele pe disc, după fiecare actualizare. Aceasta reduce riscul alterării tabelelor în eventualitatea unei căderi a sistemului, dar provoacă serioase degradări ale performanțelor. Această opțiune a fost introdusă în versiunea MySQL 3.22.9.

- `--install`

  Instalează `mysqld-nt` ca serviciu (numai pentru Windows NT).

- `--remove`

  Elimină `mysqld-nt` ca serviciu (numai pentru Windows NT).

- `--standalone`

  Pornește `mysqld_nt` ca program autonom, nu ca serviciu (numai pentru Windows NT).

Variabile pentru `mysqld`

Pentru a vedea care sunt valorile pe care `mysqld` le va utiliza în mod prestabilit, folosiți această comandă:

```
% mysqld --help
```

Pentru a vedea care sunt valorile pe care le folosește instanța serverului `mysqld` care se execută la un moment dat, utilizați această comandă:

```
% mysqladmin variables
```

De asemenea, puteți verifica valorile curente ale variabilelor emițând o interogare `SHOW VARIABLES`. Variabilele `mysqld` care pot fi configurate cu opțiunea `--set-variable` sunt prezentate în următoarea listă:

- `back_log`

  Numărul de cereri de conexiune suspendate care pot fi așezate la coadă în timp ce sunt prelucrate conexiunile actuale.

- `connect_timeout`

  Pe durata protocolului inițial de conexiune, `mysqld` va aștepta pachetele timp de `connect_timeout` secunde.

- `delayed_insert_timeout`

  Când rutina de tratare a operațiilor `INSERT DELAYED` termină de inserat rândurile așezate la coadă, așteaptă un număr de secunde egal cu valoarea acestui parametru, pentru a vedea dacă mai sosesc rânduri `INSERT DELAYED`. Dacă mai sosesc, le manipulează; dacă nu, își încheie execuția.

- `delayed_insert_limit`

  Numărul de rânduri din instrucțiunile `DELAYED INSERT` care vor fi inserate într-un tabel înainte de a verifica dacă au mai sosit alte noi instrucțiuni `SELECT` pentru tabelul respectiv. Dacă acestea au sosit, operația de inserție este suspendată pentru a permite executarea regăsirilor.

- `delayed_queue_size`

  Numărul de rânduri care pot fi așezate la coadă pentru instrucțiunile `INSERT DELAYED`. În cazul în care coada este plină, celelalte instrucțiuni `INSERT DELAYED` sunt blocate până când se creează spațiu în coadă.

- `flush_time`

  Dacă această variabilă are o valoare diferită de zero, tabelele sunt închise pentru evacuarea pe disc a modificărilor suspendate la fiecare `flush_time` secunde. Valoarea prestabilită pentru Windows este de 30 de minute; pentru a o modifica, folosiți opțiunea `--flush`.

- `join_buffer_size`

  Dimensiunea bufferului folosit la uniri complete (adică bufferul pentru uniri executate fără utilizarea indexurilor).

  Anterior versiunii MySQL 3.23, numele acestei variabile era `join_buffer`.

- `key_buffer_size`

  Dimensiunea bufferului folosit pentru blocurile de index. Acest buffer este partajat de firele rutinelor de tratare a conexiunilor.

  Anterior versiunii MySQL 3.23, numele acestei variabile era `key_buffer`.

- `long_query_time`

  Intervalul de timp, în secunde, care definește o interogare „lentă". Orice interogare care necesită un interval de timp mai mare decât acesta determină incrementarea contorului `Slow_queries`.

- `max_allowed_packet`

  Dimensiunea maximă a bufferului folosit pentru comunicarea între server și client. Inițial, dimensiunea alocată a bufferului este de `net_buffer_length` octeți, dar poate crește până la `max_allowed_packet` octeți, dacă este necesar.

- `max_connections`

  Numărul maxim de conexiuni client simultane permise.

- `max_connect_errors`

  Numărul de conexiuni ratate de la o gazdă, permise anterior blocării oricăror tentative de conectare de la gazda respectivă. Această operație se bazează pe faptul că este posibil să se încerce o pătrundere ilegală în rețea de la gazda respectivă. Pentru evacuarea zonei cache aferentă gazdelor în vederea reactivării gazdelor blocate se poate folosi instrucțiunea `FLUSH HOSTS` sau comanda `mysqladmin flush-hosts`.

- `max_delayed_insert_threads`

  Numărul maxim de rutine de tratare care vor fi inițiate pentru tratarea instrucțiunilor `INSERT DELAYED`. Toate instrucțiunile de acest gen care sunt primite în timp ce numărul maxim de rutine de tratare se află în uz vor fi tratate ca instrucțiuni ce nu sunt de tip `DELAYED`.

- `max_join_size`

  Utilitarul de optimizare MySQL estimează numărul de rânduri pe care îl va returna o operație de unire. Dacă valoarea estimată depășește `max_join_size` rânduri, va fi returnată o eroare. Această variabilă se poate folosi dacă utilizatorii au tendința de a scrie interogări `SELECT` la întâmplare, care returnează un număr aleator de rânduri.

- `max_sort_length`

  Valorile de tip `BLOB` sau `TEXT` sunt sortate folosind primii `max_sort_length` octeți ai fiecărei valori.

- `net_buffer_length`

  Dimensiunea inițială a bufferului folosit pentru comunicarea între server și client. Acest buffer poate fi extins până la un maximum de `max_allowed_packets` octeți.

- `record_buffer`

  Dimensiunea bufferului folosit de fire care execută baleieri secvențiale ale tabelelor.

- `sort_buffer`

  Dimensiunea bufferului folosit de fire pentru execuția operațiilor de sortare (`GROUP BY` sau `ORDER BY`). Această valoare corespunde unui singur fir. În mod normal, dacă puteți avea mai multe fire care execută sortarea simultan, nu este indicat ca această valoare să fie foarte mare (peste 1MB).

- `table_cache`

  Numărul maxim de tabele care pot fi deschise. Această zonă cache este partajată între fire.

- `tmp_table_size`

  Numărul maxim de octeți permis pentru tabelele temporare. Dacă un tabel depășește această dimensiune, `mysqld` returnează clientului o eroare.

- `thread_stack`

  Dimensiunea stivei pentru fiecare fir.

- `wait_timeout`

  Intervalul de timp (în secunde) în care o conexiune poate rămâne inactivă înainte ca serverul să o închidă.

mysqldump

Programul `mysqldump` scrie conținutul tabelelor din bazele de date în fișiere text. Aceste fișiere se pot folosi pentru o diversitate de scopuri, precum executarea copiilor de siguranță a bazelor de date, mutarea bazelor de date într-un alt server sau configurarea unei baze de date de test, în funcție de conținutul unei baze de date existente.

În mod prestabilit, datele de ieșire pentru fiecare tabel transferat sunt alcătuite dintr-o instrucțiune `CREATE TABLE` care re-creează tabelul, urmată de un set de instrucțiuni `INSERT` care reîncarcă datele incluse în tabel. Dacă este specificată opțiunea `--tab`, conținutul tabelului este scris într-un fișier de date în formatul brut, iar instrucțiunea SQL de creare a unui tabel este scrisă într-un fișier separat.

Utilizare

```
mysqldump [optiuni] nume_db [nume_tabel] ...
```

Dacă nu sunt menționate nume de tabele după numele bazei de date *nume_db*, vor fi transferate toate tabelele din baza de date. În caz contrar, vor fi transferate numai tabelele denumite.

Modalitatea cea mai frecventă de utilizare a programului `mysqldump` este următoarea:

```
% mysqldump --opt nume_db > nume_fisier_copie
```

Opțiuni standard acceptate de programul mysqldump

`--compress`	`--host`	`--port`	`--user`
`--debug`	`--password`	`--set-variable`	`--verbose`
`--help`	`--pipe`	`--socket`	`--version`

Opțiuni specifice programului mysqldump

Următoarele opțiuni controlează modul de operare a programului `mysqldump`. Secțiunea următoare, „Opțiuni privind formatul datelor", descrie opțiunile care se pot folosi în conjuncție cu opțiunea `--tab` pentru a indica formatul fișierelor de date.

- `--add-drop-table`

  Adaugă o instrucțiune `DROP TABLE IF EXISTS` înainte de fiecare instrucțiune `CREATE TABLE`.

- `--add-locks`

  Adaugă instrucțiunile `LOCK TABLE` și `UNLOCK TABLE` în jurul setului de instrucțiuni `INSERT` care încarcă datele pentru fiecare tabel.

- `--allow-keywords`

  Permite crearea numelor de coloană care sunt cuvinte cheie.

- `-c, --complete-insert`

  Folosește instrucțiuni `INSERT` care denumesc fiecare coloană care va fi inserată.

- `--delayed-insert`

  Scrie instrucțiuni `INSERT DELAYED`.

- `-e, --extended-insert`

  Scrie instrucțiuni `INSERT` pe mai multe rânduri.

- `-F, --flush-logs`

  Golește fișierele jurnal ale serverului înainte de a începe transferul.

- `-f, --force`

  Continuă execuția chiar dacă se produc erori.

- `--full`

  Adaugă informații suplimentare la instrucțiunile `CREATE TABLE` pe care le generează programul `mysqldump`, precum tipul tabelului, valoarea `AUTO_INCREMENT` inițială și așa mai departe. Acestea sunt informațiile pe care le puteți specifica în partea *`optiuni_tabel`* din sintaxa instrucțiunii `CREATE TABLE` (vezi Anexa D).

- `-l, --lock-tables`

  Obține o blocare pentru toate tabelele transferate, înainte de a le transfera.

- `-t, --no-create-info`

  Nu scrie instrucțiunile `CREATE TABLE`.

- `-d, --no-data`

  Nu scrie datele din tabel.

- `--opt`

  Optimizează viteza de transfer a tabelelor și scrie un fișier de transfer care este optim pentru viteza de reîncărcare. Această opțiune activează opțiunile `--add-drop-table`, `--add-locks`, `--all`, `--extended-insert`, `--quick` și `--lock-tables`.

- `-q, --quick`

  În mod prestabilit, `mysqldump` citește în memorie întreg conținutul unui tabel și apoi îl scrie. Această opțiune determină scrierea la ieșire a fiecărui rând imediat ce acesta a fost citit de la server.

  Dacă folosiți această opțiune, nu trebuie să suspendați programul `mysqldump`; astfel, veți determina serverul să aștepte, ceea ce afectează funcționarea altor clienți.

- `-T` *`dir_transfer`*, `--tab=`*`dir_transfer`*

  Această opțiune determină programul `mysqldump` să scrie două fișiere pentru fiecare tabel, folosind *`dir_transfer`* ca locație a fișierelor. Catalogul respectiv trebuie să existe deja. Pentru fiecare tabel *`nume_tabel`*, este scris un fișier *`dir_transfer/nume_tabel`*`.txt`, care conține datele din tabel, precum și un fișier *`dir_transfer/nume_tabel`*`.sql`, care conține instrucțiunea `CREATE TABLE` pentru tabel. Pentru a folosi această opțiune, trebuie să dispuneți de privilegiul `FILE`.

Pentru a evita confuzia, cel mai bine este să rulați programul `mysqldump` pe gazda serverului atunci când folosiți această opțiune. În caz contrar, unele dintre fișiere sunt scrise la server și unele la client. *`dir_transfer`* este folosit pe gazda serverului pentru fișierele `*.sql`, respectiv pe gazda clientului pentru fișierele `*.txt`. Fișierele `*.sql` vor fi în posesia contului folosit pentru rularea serverului, iar fișierele `*.txt` vor fi în posesia dumneavoastră.

În mod prestabilit, fișierul de date conține linii care se termină prin caractere linie nouă și care sunt alcătuite din valori ale coloanelor separate prin tabulatori. Acest format se poate modifica folosind opțiunile descrise în secțiunea „Opțiuni privind formatul datelor".

- `-w clauza_unde, --where=clauza_unde`

  Transferă numai înregistrările selectate de clauza `WHERE` dată de *`clauza_unde`*. Trebuie să delimitați clauza între ghilimele, pentru a împiedica shell-ul să o interpreteze sub formă de mai multe argumente în linia de comandă. Această opțiune a fost introdusă în versiunea MySQL 3.22.7.

Opțiuni privind formatul datelor

Dacă specificați opțiunea `--tab` sau `-T` în vederea generării unui fișier de date separat pentru fiecare tabel, se aplică mai multe opțiuni suplimentare. Poate că va fi necesar să delimitați valoarea opțiunii folosind caracterele de citare adecvate. Aceste opțiuni sunt analoge opțiunilor privind formatul datelor specifice instrucțiunii `LOAD DATA`. Consultați informațiile aferente instrucțiunii `LOAD DATA` în Anexa D.

- `--fields-enclosed-by=caracter`

  Specifică faptul că valorile din coloane trebuie să fie delimitate folosind caracterul respectiv, de regulă un caracter de citare. În mod prestabilit, nu se folosesc nici un fel de semne de citare.

- `--fields-escaped-by=caracter`

  Precizează caracterul escape pentru modificarea semnificației caracterelor speciale. În mod prestabilit, nu se folosește un asemenea caracter.

- `--fields-optionally-enclosed-by=caracter`

  Specifică faptul că valorile din coloane trebuie delimitate folosindu-se caracterul dat, de obicei un caracter de citare. Caracterul este folosit pentru coloane care nu conțin valori numerice. În mod prestabilit, nu se folosește un asemenea caracter.

- `--fields-terminated-by=caracter`

  Precizează caracterul de separare între valorile coloanelor utilizat în fișierele de date. În mod prestabilit, valorile sunt separate prin caractere de tip linie nouă.

- `--lines-terminated-by=sir`

  Specifică șirul (pot fi mai multe caractere) de scris la sfârșitul liniilor care conțin datele de ieșire. În mod prestabilit, sunt scrise caractere linie nouă.

Variabile pentru programul `mysqldump`

Următoarele variabile `mysqldump` pot fi configurate cu ajutorul opțiunii `--set-variable`:

- `max_allowed_packet`

  Dimensiunea maximă a bufferului folosit pentru comunicațiile între server și client.

- `net_buffer_length`

  Dimensiunea inițială a bufferului folosit pentru comunicațiile între server și client. Acest buffer poate fi extins până la valoarea maximă de `max_allowed_packet` octeți.

`mysqlimport`

Utilitarul `mysqlimport` este un utilitar de încărcare de mare capacitate pentru citirea conținutului fișierelor text în tabelele existente. Funcționează ca interfață în linie de comandă pentru instrucțiunea SQL `LOAD DATA` și constituie o modalitate eficientă de introducere a rândurilor în tabele.

Utilizare

```
mysqlimport [optiuni] nume_db nume_fisier ...
```

Argumentul *`nume_db`* specifică baza de date care conține tabelele unde doriți să încărcați date. Tabelele de încărcat sunt determinate din argumentele nume de fișier. Pentru fiecare nume de fișier, orice extensie care începe de la primul punct inclus în numele de fișier este eliminată, iar numele de bază rămas este folosit ca nume al tabelului în care trebuie încărcat fișierul. De exemplu, `mysqlimport` va încărca conținutul fișierului `presedinte.txt` în tabelul `presedinte`.

Opțiuni standard acceptate de `mysqlimport`

`--compress`	`--host`	`--port`	`--user`
`--debug`	`--password`	`--silent`	`--verbose`
`--help`	`--pipe`	`--socket`	`--version`

Opțiuni specifice programului `mysqlimport`

Următoarele opțiuni controlează modul în care programul `mysqlimport` prelucrează fișierele de intrare. Următoarea secțiune, „Opțiuni privind formatul datelor", descrie opțiuni care se pot folosi pentru a indica formatul datelor din fișierele de intrare.

- `-d, --delete`

  Golește fiecare tabel înainte de a încărca date în acesta.

- `-f, --force`

  Continuă încărcarea rândurilor, chiar dacă se produc erori.

- `-i, --ignore`

  Când un rând de intrare conține o valoare a unei chei unice deja existentă în tabel păstrează rândul existent și ignoră rândul de intrare.

- `-L, --local`

  În mod prestabilit, programul `mysqlimport` permite serverului să citească fișierul de date, ceea ce înseamnă că fișierul trebuie să se găsească pe gazda serverului și că dumneavoastră trebuie să dispuneți de privilegiul `FILE`. Dacă specificați opțiunea `--local`, indicați programului `mysqlimport` să citească personal fișierul de date și să-l trimită serverului. Acest procedeu este mai lent, dar funcționează atunci când rulați `mysqlimport` pe un alt calculator decât gazda serverului, dar și când rulați programul respectiv pe gazda serverului, chiar dacă nu aveți privilegiul `FILE`.

  Opțiunea `--local` a fost introdusă în versiunea MySQL 3.22.15.

- `-l, --lock-tables`

  Blochează fiecare tabel înainte de a introduce date în acesta.

- `-r, --replace`

  Când un rând de intrare conține o valoare a unei chei unice deja existentă în tabel, înlocuiește rândul existent cu rândul de intrare.

Opțiuni privind formatul datelor

În mod prestabilit, `mysqlimport` presupune că fișierele de date conțin linii terminate prin caractere linie nouă și alcătuite din valori separate prin tabulatori. Formatul așteptat poate fi modificat folosind următoarele opțiuni. Poate că este necesar să delimitați valoarea opțiunii între caracterele de citare adecvate. Aceste opțiuni sunt analoge opțiunilor privind formatul datelor caracteristice instrucțiunii `LOAD DATA`. Consultați informațiile aferente instrucțiunii `LOAD DATA` în Anexa D.

- `--fields-enclosed-by=`*`caracter`*

  Specifică faptul că valorile din coloane trebuie să fie delimitate folosind caracterul respectiv, de regulă un caracter de citare. În mod prestabilit, se presupune că valorile nu sunt delimitate prin nici un fel de semne de citare.

- `--fields-escaped-by=`*`caracter`*

  Precizează caracterul escape folosit pentru modificarea semnificației caracterelor speciale. În mod prestabilit, nu există un asemenea caracter.

- `--fields-optionally-enclosed-by=`*`caracter`*

  Specifică faptul că valorile din coloane trebuie delimitate folosindu-se caracterul dat, de obicei un caracter de citare.

- `--fields-terminated-by=`*`caracter`*

  Precizează caracterul de separare între valorile coloanelor. În mod prestabilit, valorile sunt separate prin caractere de tip linie nouă.

- `--lines-terminated-by=`*`sir`*

  Specifică șirul (pot fi mai multe caractere) situat la sfârșitul liniilor care conțin datele de intrare. În mod prestabilit, se presupune că liniile respective se termină prin caractere linie nouă.

mysqlshow

Utilitarul `mysqlshow` afișează bazele de date, tabelele dintr-o bază de date, respectiv informații despre coloanele sau indexurile dintr-un tabel. Se comportă ca o interfață în linie de comandă pentru instrucțiunea SQL `SHOW`.

Utilizare

```
mysqlshow [optiuni] [nume_db [nume_tabel [nume_coloana]]]
```

Dacă nu este specificat numele unei baze de date, `mysqlshow` afișează toate bazele de date din gazda serverului. Dacă este dat numele unei baze de date, dar nu este specificat numele unui tabel, vor fi afișate toate tabelele din baza de date. Dacă sunt specificate numele bazei de date și numele tabelului, dar nu este specificat numele unei coloane, afișează coloanele din tabel. Dacă sunt specificate toate numele, utilitarul `mysqlshow` afișează informații despre coloana indicată.

Dacă argumentul final conține un caracter de înlocuire pentru interpretor (`*` sau `?`), datele de ieșire sunt limitate la valorile care corespund modelului creat prin utilizarea caracterului de înlocuire. Cele două caractere menționate mai sus sunt tratate ca și caracterele de înlocuire SQL `%` și `_` aferente operatorului `LIKE`.

Opțiuni standard acceptate de mysqlshow

`--compress`	`--host`	`--port`	`--version`
`--debug`	`--password`	`--sockett`	
`--help`	`--pipe`	`--user`	

Opțiuni specifice programului mysqlshow

- `-i, --status`

  Afișează aceleași categorii de informații despre tabel ca și cele afișate de instrucțiunea `SHOW TABLE STATUS`.

  Opțiunea `--status` a fost introdusă în versiunea MySQL 3.23.

- `-k, --keys`

  Afișează informații despre indexurile din tabel, în afară de informațiile despre coloanele din tabele. Această opțiune are sens numai dacă specificați un nume de tabel.

safe_mysqld

`safe_mysqld` este un script care pornește serverul `mysqld` și îl monitorizează. Dacă serverul își încheie accidental execuția, scriptul `safe_mysqld` îl repornește.

Utilizare

Scriptul `safe_mysqld` acceptă aceleași opțiuni ca și `mysqld`.

ANEXA F

Referință API C

Această anexă descrie interfața de programare a aplicațiilor în limbajul C pentru biblioteca client MySQL. Interfața API constă dintr-un set de funcții pentru comunicarea cu serverele MySQL și obținerea accesului la bazele de date, precum și un set de tipuri de date folosite de aceste funcții.

Această anexă servește drept referință, deci include numai scurte fragmente de programe care ilustrează utilizarea bibliotecilor client. Pentru programele client complete și note cu privire la redactarea lor, examinați capitolul 6, „Interfața API MySQL pentru C".

Compilare și legare

La nivelul sursă, biblioteca client este definită în fișierul antet `mysql.h`, care poate fi inclus în propriile dumneavoastră fișiere sursă:

```
#include mysql.h
```

Pentru a arăta compilatorului unde să găsească acest fișier, poate că este necesar să specificați o opțiune `-I/cale/spre/dir/include`. De exemplu, dacă fișierele dumneavoastră antet MySQL sunt instalate în catalogul `usr/include/mysql` sau `/usr/local/mysql/include`, puteți compila fișierele sursă folosind comenzi care se prezintă astfel:

```
% gcc -I/usr/include/mysql -c progmeu.c
% gcc -I/usr/local/mysql/include -c progmeu.c
```

Dacă trebuie să folosiți alte fișiere antet MySQL, acestea sunt localizate în același catalog ca și `mysql.h`. De exemplu, `mysql_com.h` conține constante și macroinstrucțiuni pentru interpretarea metadatelor aferente rezultatelor interogării. (`mysql_com.h` este inclus în `mysql.h`, deci, deși puteți dori să examinați acest fișier pentru a vedea care este conținutul acestuia, aveți acces la fișierul respectiv prin simpla includere a fișierului `mysql.h`.) Fișierele antet `errmsg.h` și `mysqld_error.h` conțin constante pentru coduri de eroare.

La nivelul obiect, biblioteca client este furnizată sub forma bibliotecii `mysqlclient`. Pentru a lega această bibliotecă în programul dumneavoastră, specificați opțiunea `-lmysqlclient` în comanda de legare. Probabil că va trebui să indicați programului de legare unde să găsească biblioteca folosind o opțiune `-L/cale/spre/cat/bibl`. De exemplu:

```
% gcc -o progmeu progmeu.o -L/usr/lib/mysql -lmysqlclient
% gcc -o progmeu progmeu.o -L/usr/local/mysql/lib -lmysqlclient
```


Tipuri de date din interfața API C

Tipurile de date pentru biblioteca client MySQL sunt proiectate astfel încât să reprezinte entitățile pe care le utilizați în cursul unei sesiuni cu serverul. Există tipuri pentru conexiunea în sine, pentru rezultatele unei interogări, pentru un rând din cadrul unui rezultat, precum și pentru metadate (informații descriptive despre coloanele care compun un rezultat).

În expunerea următoare, noțiunile de „coloană" și „câmp" sunt sinonime.

Tipuri de date scalare

Tipurile de date scalare din MySQL reprezintă valori precum întregi de foarte mari dimensiuni, valori booleene și deplasamente de câmp.

- `my_ulonglong`

  Un tip de întreg lung, folosit pentru valorile returnate de funcții precum `mysql_affected_rows()`, `mysql_num_rows()` și `mysql_insert_id()`, care returnează numere de rânduri sau alte numere de dimensiuni mari.

  Pentru a afișa o valoare `my_ulonglong`, convertiți-o la `unsigned long` și folosiți formatul `%lu`. Dacă nu procedați astfel, valoarea nu va fi afișată corect pe unele sisteme.

- `my_bool`

  Un tip boolean, folosit pentru valorile returnate de funcțiile `mysql_change_user()` și `mysql_eof()`.

- `MYSQL_FIELD_OFFSET`

  Acest tip de date este folosit de funcțiile `mysql_field_tell()` și `mysql_field_seek()` pentru reprezentarea deplasamentelor din setul de structuri `MYSQL_FIELD` aferente setului de rezultate curent.

Tipuri de date non-scalare

Tipurile non-scalare folosite în MySQL reprezintă structuri sau tablouri. Fiecare structură `MYSQL` și `MYSQL_RES` trebuie asimilată cu o „cutie neagră". Cu alte cuvinte, trebuie să faceți referire numai la structura în sine, nu la elementele din structură. Tipurile `MYSQL_ROW` și `MYSQL_FIELD` nu prezintă o asemenea restricție; acestea sunt accesibile direct pentru obținerea datelor și a metadatelor returnate ca rezultat al unei interogări.

- `MYSQL`

  Principalul tip de bibliotecă client este structura `MYSQL`, care este folosită pentru rutinele de tratare a conexiunilor. O rutină de tratare conține informații cu privire la starea unei conexiuni din cadrul unui server. Dumneavoastră inițializați o structură `MYSQL` și apoi o transferați unei rutine de conexiune pentru a deschide o sesiune server. După ce ați stabilit conexiunea, puteți folosi rutina de tratare pentru emiterea interogărilor, pentru a genera seturi de rezultate, pentru a obține informații referitoare la erori și așa mai departe.

- `MYSQL_RES`

  Interogări precum `SELECT` sau `SHOW` returnează datele la client prin intermediul unui set de rezultate, reprezentat sub forma unei structuri `MYSQL_RES`. Această structură conține informații cu privire la rândurile returnate de interogare.

După ce ați obținut un set de rezultate, puteți apela funcții API pentru a obține datele din setul de rezultate (valorile datelor din fiecare rând al setului) sau metadate (informații despre rezultat, precum numărul de coloane, tipurile și lungimile acestora etc.)

- MYSQL_ROW

Tipul MYSQL_ROW conține valorile cuprinse într-un rând de date, reprezentat sub formă de tablou din șiruri de octeți numărate. Toate valorile sunt returnate sub formă de șir (chiar și numerele), cu excepția faptului că, dacă o valoare dintr-un rând este NULL, este reprezentată în structura MYSQL_ROW sub forma unui pointer C NULL.

Numărul de valori dintr-un rând este dat de mysql_num_fields(). Valoarea coloanei a i-a dintr-un rând este dată de row[i], unde i variază de la 0 la mysql_num_fields(res_set)-1. (res_set este un pointer spre un set de rezultate din structura MYSQL_RES.)

Rețineți că tipul MYSQL_ROW este deja un pointer, deci veți declara o variabilă de tip rând astfel:

```
MYSQL_ROW rand;        /* corect */
```

Declararea nu se va face astfel:

```
MYSQL_ROW *rand;       /* incorect */
```

Valorile dintr-un tip MYSQL_ROW se termină cu nul, deci valorile care nu sunt de tip binar pot fi tratate sub formă de șiruri care se termină cu nul. Totuși, valorile de date care pot conține date binare ar putea include intern octeți nuli și trebuie să fie tratate ca șiruri numărate. Pentru a obține lungimea valorilor din rând, apelați funcția mysql_fetch_lengths() astfel:

```
unsigned long *lungime;
lungime = mysql_fetch_lengths(res_set);
```

Lungimea returnată de mysql_fetch_lengths() pentru valorile NULL este 0.

- MYSQL_FIELD

Biblioteca client folosește structurile MYSQL_FIELD pentru a reprezenta metadate cu privire la coloanele din setul de rezultate – o structură pentru fiecare coloană. Numărul structurilor MYSQL_FIELD este dat de mysql_num_fields(). Puteți obține accesul la structuri succesive de câmpuri apelând funcția mysql_fetch_field(), respectiv vă puteți deplasa înainte și înapoi printre structuri cu mysql_field_tell() și mysql_field_seek().

Structura MYSQL_FIELD este utilă pentru prezentarea sau interpretarea conținutului rândurilor de date. Structura se prezintă astfel:

```
typedef struct st_mysql_field {
  char *name;
  char *table;
  char *def;
  enum enum_field_types type;
  unsigned int length;
  unsigned int max_length;
  unsigned int flags;
  unsigned int decimals;
} MYSQL_FIELD;
```


Membrii unei structuri MYSQL_FIELD au următoarele semnificații:

- name

  Numele coloanei, ca șir terminat cu nul. Dacă valorile coloanei sunt calculate ca rezultat al unei expresii, nume reprezintă expresia respectivă, sub formă de șir. Dacă o coloană sau o expresie primește un alias de coloană, nume este numele aliasului. De exemplu, interogarea următoare are ca rezultat valori pentru nume precum col_mea, 4*(col_mea+1) și mc:

  ```
  SELECT col_mea, 4*(col_mea+1), col_mea AS mc ...
  ```

- table

  Numele tabelului din care provine coloana, ca șir terminat cu nul. Dacă valorile din coloană sunt calculate ca rezultat al unei expresii, table este un șir vid. De exemplu, dacă emiteți o interogare precum cea indicată mai jos, numele tabelului pentru prima coloană (col_mea) este tabel_meu, în timp ce numele tabelului pentru a doua coloană (col_mea+0) este șirul vid:

  ```
  SELECT col_mea, my_col+0 FROM tabel_meu ...
  ```

- def

  Valoarea prestabilită pentru coloană, ca șir terminat cu nul. Acest membru al structurii MYSQL_FIELD este determinat numai dacă apelați funcția mysql_list_fields(), în caz contrar având valoarea NULL. (Valorile prestabilite ale coloanelor pot fi de asemenea obținute prin emiterea unei interogări DESCRIBE *nume_tabel* sau SHOW FIELDS FROM *nume_tabel* și prin examinarea setului de rezultate.)

- type

  Tipul coloanei. Tipul unei coloane calculate ca rezultat al unei expresii este determinat din tipurile elementelor expresiei. De exemplu, în cazul în care col_mea este o coloană de tip VARCHAR(20), type are valoarea FIELD_TYPE_VAR_STRING, în timp ce atributul type al valorii LENGTH(col_mea) este FIELD_TYPE_LONGLONG.

Valorile posibile ale membrului type sunt enumerate în mysql_com.h și prezentate în tabelul F.1.

Tabelul F.1 Valorile membrului type al structurii MYSQL_FIELD

Valoarea membrului type	Tipul coloanei
FIELD_TYPE_BLOB	BLOB sau TEXT
FIELD_TYPE_DATE	DATE
FIELD_TYPE_DATETIME	DATETIME
FIELD_TYPE_DECIMAL	DECIMAL sau NUMERIC
FIELD_TYPE_DOUBLE	DOUBLE sau REAL
FIELD_TYPE_ENUM	ENUM

Continuare

Tabelul F.1 Continuare

Valoarea membrului type	Tipul coloanei
FIELD_TYPE_FLOAT	FLOAT
FIELD_TYPE_INT24	MEDIUMINT
FIELD_TYPE_LONG	INT
FIELD_TYPE_LONGLONG	BIGINT
FIELD_TYPE_NULL	NULL
FIELD_TYPE_SET	SET
FIELD_TYPE_SHORT	SMALLINT
FIELD_TYPE_STRING	CHAR
FIELD_TYPE_TIME	TIME
FIELD_TYPE_TIMESTAMP	TIMESTAMP
FIELD_TYPE_TINY	TINYINT
FIELD_TYPE_VAR_STRING	VARCHAR
FIELD_TYPE_YEAR	YEAR

Puteți vedea referințe la FIELD_TYPE_CHAR în fișiere sursă mai vechi; acesta a fost un tip de un octet, care acum se numește FIELD_TYPE_TINY.

- length

  Lungimea coloanei, conform specificațiilor din instrucțiunea CREATE TABLE folosită pentru crearea tabelelor. Lungimea unei coloane calculate ca rezultat al unei expresii este determinată din elementele expresiei.

- max_length

  Lungimea valorii celei mai mari existente efectiv în setul de rezultate. De exemplu, dacă o coloană din setul de rezultate conține valorile „Nelu", „Paul" și „Constantin", valoarea max_length pentru coloană este 10.

  Deoarece valoarea max_length poate fi determinată numai după examinarea tuturor rândurilor, această valoare este semnificativă numai pentru seturile de rezultate create cu mysql_store_result(). max_length este egală cu 0 pentru seturile de rezultate create cu mysql_use_result().

- flags

  Membrul flags specifică atributele pentru coloane. Atributele sunt reprezentate prin biți individuali în cadrul valorii flags. Biții pot fi testați prin intermediul constantelor de tip mască de biți prezentate în tabelul F.2. De exemplu, pentru a determina dacă valorile unei coloane sunt sau nu de tipul UNSIGNED, testați valoarea flags astfel:

```
if (field-flags & UNSIGNED_FLAG)
  printf ("%s valori sunt UNSIGNED\n", field-name);
```


Tabelul F.2 Valorile membrului flags al structurii MYSQL_FIELD

Valoarea flags	Semnificație
AUTO_INCREMENT_FLAG	Coloana are atributul AUTO_INCREMENT
BINARY_FLAG	Coloana are atributul BINARY
MULTIPLE_KEY_FLAG	Coloana face parte dintr-un index NON-UNIQUE
NOT_NULL_FLAG	Coloana nu poate conține valori NULL
PRI_KEY_FLAG	Coloana face parte dintr-o cheie PRIMARY KEY
UNIQUE_KEY_FLAG	Coloana face parte dintr-un index UNIQUE
UNSIGNED_FLAG	Coloana are atributul UNSIGNED
ZEROFILL_FLAG	Coloana are atributul ZEROFILL

Câteva constante de tip flags indică tipuri de coloane, nu atribute de coloane; acestea nu mai sunt utilizate în prezent, deoarece trebuie să folosiți field-type pentru a determina tipul coloanei. Tabelul F.3 prezintă aceste constante ieșite din uz.

Tabelul F.3 Valori ieșite din uz ale membrului flags al structurii MYSQL_FIELD

Valoarea membrului flags	Semnificație
BLOB_FLAG	Coloana conține valori BLOB
ENUM_FLAG	Coloana conține valori ENUM
SET_FLAG	Coloana conține valori SET
TIMESTAMP_FLAG	Coloana conține valori TIMESTAMP

- decimals

  Numărul de zecimale pentru coloanele numerice; decimals este zero pentru coloanele care nu conțin valori numerice. De exemplu, o coloană DECIMAL(8,3) are o valoare decimals egală cu 3, în timp ce aceeași valoare este egală cu 0 în cazul coloanei BLOB.

Macrocomenzi accesorii

mysql.h conține câteva macrocomenzi care vă permit să testați membrii structurii MYSQL_FIELDS într-un mod mai convenabil. Funcția IS_NUM() testează membrul type; celelalte funcții prezentate aici testează membrul flags.

IS_NUM() este adevărat (diferit de zero) dacă valorile din coloană sunt de tip numeric:

```
if (IS_NUM (field->type))
   printf("Campul %s este numeric\n", field->name);
```

IS_PRI_KEY() are valoarea „adevărat" dacă o coloană face parte dintr-o cheie primară:

```
if (IS_PRI_KEY (field->flags))
   printf("Campul %s face parte dintr-o cheie primara\n", field->name);
```

IS_NOT_NULL() are valoarea „adevărat" dacă o coloană nu poate conține valori NULL:

```
if (IS_NOT_NULL (field->flags))
   printf("Valorile din campul %s nu pot fi NULL\n", field->name);
```

Funcția IS_BLOB() are valoarea „adevărat" în cazul în care coloana este de tip BLOB sau TEXT. Cu toate acestea, macrocomanda testează bitul membrului flags BLOB_FLAG scos din uz, deci și IS_BLOB() este scoasă din uz.

Funcții ale interfeței API C

Funcțiile din biblioteca client pot fi clasificate în următoarele categorii:

- Rutine de gestionare a conexiunii, pentru stabilirea și terminarea conexiunilor cu serverul
- Rutine de stare și de raportare a erorilor pentru obținerea de mesaje și coduri de eroare
- Rutine de construcție și execuție a interogărilor pentru construirea interogărilor și trimiterea acestora la server
- Rutine de prelucrare a seturilor de rezultate, pentru tratarea rezultatelor interogărilor care returnează date
- Rutine de informare, care furnizează informații despre client, server, versiunea protocolului și conexiunea curentă
- Rutine de administrare pentru controlul modului de operare a serverului
- Rutine de depanare, pentru generarea informațiilor de depanare
- Rutine scoase din uz, care acum sunt considerate depășite

Funcțiile din biblioteca client aferente fiecăreia dintre aceste categorii sunt descrise detaliat în secțiunile următoare, în ordine alfabetică, în cadrul fiecărei categorii. Anumiți parametri apar de mai multe ori în descrierile funcțiilor și au o semnificație standard:

- conn este un pointer spre variabila MYSQL de tratare a conexiunilor pentru o conexiune cu serverul.
- res_set este un pointer spre o structură cu seturi de rezultate MYSQL_RES.
- field este un pointer spre o structură MYSQL_FIELD cu informații privind coloanele.
- row este un rând de date MYSQL_ROW dintr-un set de rezultate.

Din motive de concizie, acolo unde acești parametri nu sunt menționați în descrierea funcțiilor în care apar, puteți lua în considerare descrierile menționate anterior.

Dacă nu este specificat altceva, puteți presupune că orice funcție dată există în biblioteca client începând de la versiunea MySQL 3.21.0.

Rutine de gestionare a conexiunii

Aceste funcții vă permit să stabiliți și să terminați conexiuni cu un server, să stabiliți opțiuni care influențează modul de stabilire a conexiunii, precum și să restabiliți conexiuni expirate.

- my_bool

  mysql_change_user (MYSQL *conn, char *user_name, char*password, char *db_name);

  Modifică utilizatorul și baza de date prestabilită ale conexiunii specificate de conn. Baza de date devine prestabilită pentru referințele la tabele care nu includ un specificator pentru baza de date. Dacă db_name este NULL, nu este selectată nici o bază de date prestabilită.

mysql_change_user() returnează „adevărat" dacă utilizatorul are permisiunea de a se conecta la server și, dacă a fost specificată o bază de date, de a obține accesul la baza de date. În caz contrar, funcția eșuează și utilizatorul și baza de date curente rămân nemodificate.

Funcția mysql_change_user() a fost introdusă în versiunea MySQL 3.23.3.

- void

  mysql_close(MYWQL *conn);

  Închide conexiunea specificată de conn. Apelați această rutină când ați terminat o sesiune server. În cazul în care variabila de tratare a conexiunii a fost alocată automat de către mysql_init(), funcția mysql_close() anulează alocarea.

  Nu apelați funcția mysql_close() dacă încercarea de a deschide o conexiune a eșuat.

- MYSQL *

  mysql_init(MYSQL *conn);

  Inițializează o variabilă de tratare a conexiunilor și returnează un pointer la aceasta. În cazul în care conn indică spre o structură MYSQL existentă, mysql_init() inițializează acea variabilă și returnează adresa acesteia:

```
MYSQL conn_struct, *conn;
conn = mysql_init (&conn_struct);
```

  În cazul în care conn este NULL, mysql_init() alocă o nouă variabilă de tratare, o inițializează și îi returnează adresa:

```
MYSQL *conn;
conn = mysql_init (NULL);
```

  Dacă funcția mysql_init() eșuează, returnează NULL. Aceasta se poate întâmpla dacă mysql_init() nu poate aloca o nouă variabilă de tratare.

  Dacă mysql_init() alocă variabila de tratare, mysql_close() anulează automat alocarea acesteia atunci când închideți conexiunea.

  Funcția mysql_init() a fost introdusă în MySQL versiunea 3.22.1.

- int

  mysql_options(MYSQL *conn, enum mysql_option option, char *arg);

  Această funcție vă permite să ajustați comportarea conexiunii cu mai multă precizie decât folosind numai funcția mysql_real_connect(). Apelați această funcție după mysql_init() și înainte de mysql_real_connect(). Puteți apela funcția mysql_options() de mai multe ori, dacă doriți să configurați mai multe opțiuni.

  Argumentul option specifică opțiunea de conexiune pe care doriți să o configurați. Informațiile suplimentare necesare pentru a configura opțiunea, dacă există, sunt specificate de argumentul arg. (Rețineți că arg este un pointer.) arg este NULL dacă nu sunt necesare informații suplimentare.

  Sunt disponibile următoarele opțiuni:

- MYSQL_INIT_COMMAND

  Specifică o interogare care se va executa după conectarea la server. arg este un șir terminat cu nul care conține interogarea. Interogarea este executată și după reconectare (de exemplu, dacă apelați funcția mysql_ping()). Orice set de rezultate returnat de interogare este ignorat.

- MYSQL_OPT_COMPRESS

  Specifică faptul că respectiva conexiune trebuie să utilizeze protocolul client/server comprimat, dacă serverul îl acceptă. arg este NULL.

  De asemenea, se poate specifica utilizarea comprimării atunci când apelați funcția mysql_real_connect().

- MYSQL_OPT_CONNECT_TIMEOUT

  Indică intervalul de expirare a conexiunii, exprimat în secunde. arg este un pointer spre un întreg fără semn care conține valoarea de expirare.

- MYSQL_OPT_NAMED_PIPE

  Indică faptul că acea conexiune la server trebuie să folosească un canal denumit. arg este NULL. Această opțiune este destinată numai clienților Windows 95/98/NT și numai pentru conexiunile la serverele Windows NT.

- MYSQL_READ_DEFAULT_FILE

  Specifică un fișier cu opțiuni care va fi citit pentru determinarea parametrilor de conexiune. Opțiunile sunt citite din grupul [client] din fișier.

- MYSQL_READ_DEFAULT_GROUP

  Specifică un grup care se va citi din fișierul cu opțiuni denumit cu opțiunea MYSQL_READ_DEFAULT_FILE. Grupul denumit este citit alături de grupul [client]. Dacă nu este denumit nici un fișier cu opțiuni, biblioteca client caută fișierele cu opțiuni standard și le citește.

  Dacă apelați mysql_options() de mai multe ori pentru a configura o opțiune dată, mysql_real_connect() folosește valoarea specificată cel mai recent pentru opțiunea respectivă.

  Funcția mysql_options() returnează zero în caz de reușită, respectiv o valoare diferită de zero dacă valoarea option este necunoscută.

  Într-un fișier cu opțiuni se pot folosi următoarele opțiuni (în grupul [client] sau într-un grup pe care-l specificați cu opțiunea MYSQL_READ_DEFAULT_GROUP):

```
compress
database=nume_db
debug
host=nume_gazda
init-command=interogare
password=parola_dv
pipe
port=numar_port
return-found-rows
socket=nume_soclu
timeout=secunde
user=nume_utilizator
```

  Apelurile la funcția mysql_options() din exemplul următor configurează opțiunile de conexiune astfel încât mysql_real_connect() se conectează prin intermediul unui canal denumit, folosind protocolul comprimat și un interval de expirare de 10 secunde.

  Pentru a obține informații de la grupurile [client] și [mygroup], funcția citește fișierul C:\my.cnf.special. La stabilirea conexiunii, se va executa o instrucțiune SET SQL_BIG_TABLES.

```
MYSQL *conn;
unsigned int expirare;

if ((conn = mysql_init (NULL)) == NULL)
        ... tratarea erorii ...
expirare = 10;
mysql_options (conn, MYSQL_OPT_CONNECT_TIMEOUT, (char *) &timeout);
mysql_options (conn, MYSQL_OPT_COMPRESS, NULL);
mysql_options (conn, MYSQL_NAMED_PIPE, NULL);
mysql_options (conn, MYSQL_READ_DEFAULT_FILE, "C:\my.cnf.special");
mysql_options (conn, MYSQL_READ_DEFAULT_GROUP, "grupmeu");
mysql_options (conn, MYSQL_INIT_COMMAND, "SET SQL_BIG_TABLES=1");
if (mysql_real_connect (conn, ...) == NULL)
        ... tratarea erorii ...
```

  Funcția mysql_options() a fost introdusă în MySQL 3.22.1.

  MYSQL_INIT_COMMAND, MYSQL_READ_DEFAULT_FILE și MYSQL_READ_DEFAULT_GROUP au fost introduse în versiunea MySQL 3.22.10.

- int

  mysql_ping(MYSQL *conn);

  Verifică dacă conexiunea indicată de conn este încă funcțională. Dacă nu este, funcția mysql_ping() se conectează din nou, folosind aceiași parametri care au fost utilizați inițial pentru stabilirea conexiunii. Astfel, nu trebuie să apelați funcția mysql_ping() fără a fi apelat cu succes, în prealabil, funcția mysql_real_connect().

  Funcția mysql_ping() a fost introdusă în versiunea MySQL 3.22.1.

- MYSQL *

```
mysql_real_connect (MYSQL *conn, char *nume_gazda,
                    char *nume_utilizator, char *parola,
                    char *nume_db, unsigned int numar_port,
                    char *nume_soclu, unsigned int flags);
```

  Se conectează la un server și returnează un pointer spre variabila de tratare a conexiunii. conn trebuie să fie un pointer spre o variabilă existentă de tratare a conexiunii care a fost inițializată de mysql_init(). Adresa variabilei de tratare este valoarea returnată în caz de conexiune reușită. Dacă se produce o eroare, funcția returnează NULL.

  În cazul în care tentativa de conexiune eșuează, puteți transmite valoarea variabilei de tratare conn funcțiilor mysql_errno() și mysql_error() pentru a obține informații despre eroare. Totuși, nu trebuie să transferați valoarea conn nici unei alte rutine din biblioteca client care presupune că variabila de tratare reflectă o conexiune reușită.

  nume_gazda este serverul la care se va stabili conexiunea. Tabelul F.4 prezintă modul în care clientul încearcă să se conecteze pentru diferite valori nume_gazda pentru clienți

UNIX și Windows. (Dacă vă conectați folosind un soclu UNIX sau un canal denumit, parametrul `nume_soclu` specifică numele soclului sau al canalului.)

Tabelul F.4 Tipuri de conexiune client, în funcție de tipul numelui gazdei serverului

Numele gazdei	Tip de conexiune UNIX	Tip de conexiune Windows
`nume_gazda`	Conexiune TCP/IP cu gazda denumită	Conexiune TCP/IP cu gazda denumită
Număr IP	Conexiune TCP/IP cu gazda denumită	Conexiune TCP/IP cu gazda denumită
`localhost`	Conexiune prin soclu UNIX cu gazda locală	Conexiune TCP/IP cu gazda denumită.
.	`Nu este valabil`	`Conexiune prin canal denumit cu gazda locală`
`NULL`	Conexiune prin soclu UNIX la gazda locală	Conexiune TCP/IP la gazda locală, cu excepția că în Windows NT se încearcă mai întâi o conexiune prin canal denumit înainte de a se recurge la TCP/IP

`nume_utilizator` este numele dumneavoastră de utilizator MySQL. Dacă acesta este `NULL`, biblioteca client trimite un nume prestabilit. Sub UNIX, numele prestabilit este numele dumneavoastră de deschidere a sesiunii de lucru. Sub Windows, numele prestabilit este numele dumneavoastră așa cum este specificat în variabila de mediu `USER`, dacă este activată, respectiv `"ODBC"` în caz contrar.

`parola` este parola dumneavoastră. Dacă este `NULL`, vă veți putea conecta numai dacă tabelul de acordare `user` conține o intrare care corespunde numelui dumneavoastră de utilizator și gazdei de la care vă conectați și dacă intrarea respectivă conține o parolă vidă.

`nume_db` este baza de date care se va utiliza. Dacă este `NULL`, nu va fi selectată nici o bază de date inițială.

`numar_port` este numărul portului folosit pentru conexiunile TCP/IP. Dacă este `0`, se va folosi numărul de port prestabilit.

Numărul portului și numele soclului sunt folosite în conformitate cu valoarea parametrului `nume_gazda`. Dacă vă conectați la gazda locală, funcția `mysql_real_connect()` încearcă o conexiune folosind un soclu de domeniu UNIX (sub UNIX), respectiv un canal denumit (sub Windows). Altfel, se conectează folosind TCP/IP.

`flags` poate fi `0` pentru a nu specifica nici o opțiune, respectiv poate lua una dintre valorile prezentate în tabelul F.5. Opțiunile influențează modul de operare a serverului.

Tabelul F.5 Valorile `flags` pentru funcția `mysql_real_connect()`

Valoarea indicatorului	Efectul asupra modului de operare a serverului
`CLIENT_FOUND_ROWS`	Pentru interogările `UPDATE`, returnează numărul de rânduri care au corespuns criteriilor de căutare, nu numărul rândurilor modificate.

Valoarea indicatorului	Efectul asupra modului de operare a serverului
`CLIENT_NO_SCHEMA`	Nu permite sintaxa *`nume_db.nume_tabel.nume_coloana`*
`CLIENT_COMPRESS`	Folosește protocolul de comunicații comprimat, dacă serverul îl acceptă.
`CLIENT_ODBC`	Tratează clientul ca pe un client ODBC.

Dacă specificați `CLIENT_NO_SCHEMA`, serverul permite utilizarea în interogări numai a referințelor de tipul *`nume_tabel.nume_coloana`*, *`nume_tabel`* sau *`nume_coloana`*.

Valorile indicatorilor reprezintă valori de biți, deci le puteți combina folosind caracterele | sau +; de exemplu `CLIENT_COMPRESS|CLIENT_ODBC` sau `CLIENT_COMPRESS+CLIENT_ODBC`.

Funcția `mysql_real_connect()` a fost introdusă în versiunea MySQL 3.21.10. Parametrul `nume_db` a fost adăugat în versiunea 3.22.0. Utilizarea funcției `mysql_init()` pentru inițializarea argumentului `MYSQL` a început de la versiunea MySQL 3.22.1.

Funcții de stare și de raportare a erorilor

Funcțiile din această secțiune vă permit să determinați și să raportați cauzele erorilor:

- `unsigned int`

  `mysql_errno``(MYSQL *conn);`

  Returnează un cod de eroare pentru rutina din biblioteca client cel mai recent invocată care a returnat o stare. Codul de eroare este `0` dacă nu s-a produs nici o eroare, respectiv un cod diferit de zero în caz contrar. Codurile de eroare posibile sunt enumerate în fișierele antet MySQL `errmsg.h` și `mysqld.h`.

```
if (mysql_errno(conn) = = 0)
        printf("Totul este in regula\n");
else
        printf("S-a intamplat ceva!\n");
```

  Funcția `mysql_errno()` a fost introdusă în versiunea MySQL 3.21.7.

- `char *`

  `mysql_error``(MYSQL *conn);`

  Returnează un șir terminat cu nul care conține un mesaj de eroare pentru rutina din biblioteca client cel mai recent invocată care a returnat o stare. Valoarea returnată este șirul vid dacă nu s-a produs nici o eroare (este șirul de lungime zero `""`, nu un pointer `NULL`). Deși în mod normal apelați funcția `mysql_error()` după ce ați aflat deja că s-a produs o eroare, se poate folosi chiar valoarea returnată pentru a detecta apariția unei erori:

```
if (mysql_errno(conn)[0] = = '\0') /* sirul este vid? */
        printf("Totul este in regula\n");
else
        printf("S-a intamplat ceva!\n");
```

  Funcția `mysql_error()` a fost introdusă în versiunea MySQL 3.21.7 (poate ceva mai târziu).

Rutine de construcție și execuție a interogărilor

Funcțiile din această secțiune vă permit să trimiteți interogări serverului.

`mysql_escape_string()` vă ajută să construiți interogări prin modificarea semnificației anumitor caractere care necesită un tratament special.

- `unsigned int`

  `mysql_escape_string``(char *to_str, char *from_str, unsigned int from_len);`

  Codifică un șir care poate conține caractere speciale, astfel încât să poată fi folosit într-o instrucțiune SQL. Tabelul F.6 enumeră caracterele care sunt considerate speciale și codificările acestora.

Tabelul F.6 Codificări de caractere pentru funcția `mysql_escape_string()`

Caracter special	Codificare
NUL (ASCII 0)	\0 (backslash-0, nu backslash-nul)
Backslash	\\ (backslash-backslash)
Ghilimele simple	\' (backslash-ghilimele simple)
Ghilimele duble	\" (backslash-ghilimele duble)
Linie nouă	\n (backslash-n)
Retur de car	\r (backslash-r)
Ctrl-Z	\Z (backslash-Z)

Bufferul care va fi codificat este specificat sub formă de șir numărat. `from_str` indică spre bufferul respectiv, iar `from_len` indică numărul de octeți din acesta.

Funcția `mysql_escape_string()` scrie rezultatul în bufferul indicat de `to_str` și adaugă un octet nul. `to_str` trebuie să indice spre un buffer existent, care să aibă minimum `(from_len*2)+1` octeți lungime. (În cel mai rău caz, fiecare caracter din `from_str` trebuie codificat sub forma unei secvențe de două caractere; de asemenea, mai aveți nevoie de spațiu pentru nulul de terminare.)

Funcția `mysql_escape_string()` returnează lungimea șirului codificat, fără a număra octetul nul de terminare.

Șirul codificat rezultant nu conține caractere nul interne, dar se termină cu nul, deci îl puteți folosi cu funcții precum `strlen()` sau `strcat()`.

Când scrieți șiruri literale în programul dumneavoastră, fiți atent să nu confundați convențiile lexicale de modificare a semnificației caracterelor specifice limbajului C cu codificările executate de funcția `mysql_escape_string()`. Să examinăm următorul exemplu și rezultatul determinat de acesta:

```
to_len = mysql_escape_string (to_str, "\0\\\'\"\n\r\032", 7);
printf("to_len = %d, to_str = %s/n", to_len, to_str);
```

Datele de ieșire sunt următoarele:

```
to_len = 14, to_str = \0\\\'\"\n\r\Z
```


Valoarea afișată a șirului `to_str` seamănă foarte mult cu șirul specificat ca al doilea argument al apelului la funcția `mysql_escape_string()`, dar, în realitate, este foarte diferit de acesta.

- `int`

  `mysql_query``(MYSQL *conn, char *sir_interogare);`

  Dacă este dată o interogare cu un șir care se termină cu nul, `mysql_query()` trimite interogarea serverului, pentru a fi executată. Șirul nu trebuie să conțină date binare; în particular, nu trebuie să conțină octeți nuli, pe care funcția `mysql_query()` îi interpretează ca fiind sfârșitul interogării. Dacă interogarea dumneavoastră conține date binare, folosiți în schimb funcția `mysql_real_query()`.

  Interogarea trebuie să fie alcătuită dintr-o singură instrucțiune SQL și nu trebuie să se încheie cu punct și virgulă (;) sau cu `\g`; acestea reprezintă convenții ale programului client `mysql`, nu ale bibliotecii client.

  Funcția `mysql_query()` returnează zero în caz de reușită și o valoare diferită de zero în caz de eșec. O interogare reușită este una pe care serverul o acceptă ca fiind corectă și pe care o execută fără eroare. În caz de succes, nu se menționează nimic despre numărul rândurilor afectate sau returnate.

- `int`

  `mysql_real_query``(MYSQL *conn, char *sir_interogare, unsigned int lungime);`

  Dacă este dată o interogare specificată sub formă de șir numărat, funcția `mysql_real_query()` trimite interogarea la server, pentru a fi executată. Șirul poate conține date binare (inclusiv octeți nuli). Textul interogării este dat de `sir_interogare`, iar lungimea este indicată de `lungime`.

  Interogarea trebuie să fie alcătuită dintr-o singură instrucțiune SQL și nu trebuie să se încheie cu punct și virgulă (;) sau cu `\g`; acestea reprezintă convenții ale programului client `mysql`, nu ale bibliotecii client.

  Funcția `mysql_real_query()` returnează zero în caz de reușită și o valoare diferită de zero în caz de eșec. O interogare reușită este una pe care serverul o acceptă ca fiind corectă și care se execută fără eroare. În caz de succes, nu se menționează nimic despre numărul rândurilor afectate sau returnate.

- `int`

  `mysql_select_db``(MYSQL *conn, char *nume_db);`

  Selectează baza de date denumită de `nume_db` drept bază de date curentă, care devine bază de date prestabilită pentru referințele la tabele care nu conțin nici un specificator explicit al bazei de date. Trebuie să aveți permisiunea de a obține accesul la baza de date; în caz contrar, funcția `mysql_select_db()` eșuează.

  Funcția `mysql_select_db()` returnează zero în caz de succes, respectiv o valoare diferită de zero în caz de eșec.

Rutine de prelucrare a setului de rezultate

Când o interogare produce un set de rezultate, funcțiile din această secțiune vă permit să regăsiți setul și să obțineți acces la conținutul său. Funcțiile `mysql_store_result()` și `mysql_use_result()` creează setul de rezultate, iar una din ele trebuie apelată înainte de utilizarea oricăror alte funcții din această secțiune. Tabelul F.7 compară cele două funcții.

Tabelul F.7 O comparație între funcțiile `mysql_store_result()` și `mysql_use_result()`.

mysql_store_result()	**mysql_use_result()**
Toate rândurile din setul de rezultate sunt preluate chiar de funcția `mysql_store_result()`.	`mysql_use_result()` inițializează setul de rezultate, dar lasă regăsirea rândurilor în seama funcției `mysql_fetch_row()`
Folosește o cantitate mai mare de memorie; toate rândurile sunt stocate la client.	Folosește o cantitate mai redusă de memorie; este stocat fiecare rând în parte.
Este mai lentă, datorită suprasarcinii implicate de alocarea de memorie pentru întregul set de rezultate.	Este mai rapidă, deoarece este necesară alocarea de memorie numai pentru rândul curent.
Valoarea `NULL` returnată de funcția `mysql_fetch_row()` indică sfârșitul setului de rezultate; nu indică niciodată o eroare.	Valoarea `NULL` returnată de funcția `mysql_fetch_row()` indică sfârșitul setului de rezultate sau o eroare, deoarece o pană de comunicare poate afecta regăsirea înregistrării curente.
Funcția `mysql_num_rows()` poate fi apelată în orice moment după apelarea funcției `mysql_store_result()`.	Funcția `mysql_num_rows()` returnează un număr corect de rânduri numai după preluarea tuturor rândurilor.
Funcția `mysql_affected_rows()` este sinonimă cu `mysql_num_rows()`.	Funcția `mysql_affected_rows()` nu poate fi folosită.
Accesul aleator la rândurile din setul de rezultate este posibil cu funcțiile `mysql_data_seek()`, `mysql_row_seek()` și `mysql_row_tell()`.	Nu este permis accesul aleator la setul de rezultate; rândurile trebuie prelucrate în ordinea în care sunt returnate de server; funcțiile `mysql_data_seek()`, `mysql_row_seek()` și `mysql_row_tell()` nu trebuie folosite.
Nu trebuie să apelați deloc funcția `mysql_fetch_row()`, deoarece toate rândurile au fost deja regăsite de client.	*Trebuie* să apelați funcția `mysql_fetch_row()` pentru a prelua toate rândurile; în caz contrar, rândurile „rămase" se vor infiltra în rezultatul următoarei interogări, determinând o eroare de sincronizare.
Tabelele sunt blocate la citire pentru un interval de timp strict necesar preluării rândurilor de date.	Tabelele pot rămâne în starea „blocat pentru citire" dacă clientul se oprește în mijlocul unei operații de regăsire, blocând orice client care încearcă să le modifice.
Membrul `max_length` al structurilor de seturi de rezultate `MYSQL_FIELD` primește cea mai lungă valoare existentă în setul de rezultate pentru coloanele din set.	`max_length` nu este configurat la nici o valoare semnificativă, deoarece aceasta nu poate fi cunoscută decât după regăsirea tuturor rândurilor.

- `my_ulonglong`

  `mysql_affected_rows``(MYSQL *conn);`

  Returnează numărul de rânduri modificate de cea mai recentă interogare `DELETE`, `INSERT`, `REPLACE` sau `UPDATE`. Pentru asemenea interogări, funcția `mysql_affected_rows()` poate fi apelată imediat după un apel reușit la funcția `mysql_query()`. De asemenea, puteți apela această funcție după ce ați emis o instrucțiune care returnează rânduri. În acest caz, funcția se comportă în același mod ca și `mysql_num_rows()` și este supusă aceloraşi restricții pentru care valoarea are sens, precum și la restricția suplimentară conform căreia funcția `mysql_affected_rows()` nu este niciodată semnificativă atunci când folosiți `mysql_use_result()`.

  Funcția `mysql_affected_rows()` returnează zero dacă nu a fost emisă nici o interogare sau dacă interogarea returnează rânduri, dar nu a fost selectat nici un rând. O valoare returnată mai mare decât zero indică numărul rândurilor modificate (pentru `DELETE`, `INSERT`, `REPLACE` și `UPDATE`) respectiv returnate (pentru interogările care returnează rânduri). O valoare returnată egală cu –1 indică fie o eroare, fie că ați apelat (în mod greșit) funcția `mysql_affected_rows()` după ce ați emis o interogare care returnează rânduri, dar înainte de a regăsi efectiv setul de rezultate. Deoarece funcția `mysql_affected_rows()` returnează o valoare fără semn, trebuie să efectuați comparația prin conversia rezultatului său într-o valoare cu semn:

```
if ((long) mysql_affected_rows(conn) = = -1)
        fprintf(stderr, "Eroare!\n");
```

  Dacă ați specificat că clientul trebuie să returneze numărul de rânduri care au corespuns criteriilor de căutare în cazul interogărilor `UPDATE`, funcția `mysql_affected_rows()` returnează valoarea respectivă, nu numărul rândurilor efectiv modificate. (MySQL nu actualizează un rând în cazul în care coloanele care urmează a fi modificate sunt identice cu noile valori.) Această comportare poate fi selectată fie prin specificarea opțiunii `return-found-rows` într-un fișier cu opțiuni, fie prin transferarea opțiunii `CLIENT_FOUND_ROWS` în parametrul `flags` către funcția `mysql_real_connect()`.

  Vezi nota despre afișarea valorilor de tip `my_ulonglong` din secțiunea „Tipuri de date scalare".

- `void`

  `mysql_data_seek``(MYSQL_RES *res_set, unsigned int offset);`

  Caută într-un anumit rând din setul de rezultate. Valoarea parametrului `offset` poate varia între `0` și `mysql_num_rows(res_set)-1`. Rezultatele nu pot fi anticipate dacă `offset` nu se găsește în acest domeniu.

  `mysql_data_seek()` impune ca întregul set de rezultate să fie regăsit, deci puteți folosi această funcție numai dacă setul de rezultate a fost creat cu funcția `mysql_store_result()`, nu cu funcția `mysql_use_result()`.

- `MYSQL_FIELD *`

  `mysql_fetch_field``(MYSQL_RES *res_set);`

  Returnează o structură care conține informații (metadate) despre o coloană din setul de rezultate. După ce ați executat cu succes o interogare care returnează rânduri,

primul apel la funcția mysql_fetch_field() returnează informații despre prima coloană. Apelurile ulterioare returnează informații despre coloanele următoare, respectiv NULL când nu a mai rămas nici o coloană.

De asemenea, puteți apela funcția mysql_field_tell() pentru a determina poziția coloanei curente, respectiv funcția mysql_field_seek() pentru a selecta o anumită coloană care va fi returnată de următorul apel la funcția mysql_fetch_field().

Exemplul următor caută prima structură MYSQL_FIELD, apoi preia următoarele structuri de informații privind coloanele:

```
MYSQL_FIELD          *field;
unsigned int         i;

mysql_field_seek(res_set, 0);
for (i = 0; i < mysql_num_fields(res_set); i++)
{
        field = mysql_fetch_field(res_set);
        printf("column %u: name = %s max_length = %lu\n",
                i, field->name, field->max_length);
}
```

- MYSQL_FIELD *

mysql_fetch_fields(MYSQL_RES res_set);

Returnează un tablou cu toate structurile care conțin informații despre coloane aferente setului de rezultate. La acestea se poate obține accesul astfel:

```
MYSQL_FIELD          *field;
unsigned int         i;

field = mysql_fetch_fields(res_set);
for (i = 0; i < mysql_num_fields(res_set); i++)
{
        printf("column %u: name = %s max_length = %lu\n",
                i, field[i].name, field[i].max_length);
}
```

Comparați acest listing cu exemplul prezentat pentru funcția mysql_fetch_field(). Rețineți că, deși ambele funcții returnează valori de același tip, aceste valori sunt accesibile folosind sintaxe ușor diferite pentru fiecare funcție în parte.

- MYSQL_FIELD *

mysql_fetch_field_direct(MYSQL_RES *res_set, unsigned int field_num);

Dacă este dat indexul unei coloane, returnează structura de informații aferentă acelei coloane. Valoarea parametrului field_num poate varia între 0 și mysql_num_fields()-1. Rezultatele sunt imprevizibile dacă field_num nu se încadrează în domeniu.

În exemplul următor, se obține accesul direct la structurile MYSQL_FIELD:

```
MYSQL_FIELD          *field;
unsigned int         i;
```



```
for (i = 0; i < mysql_num_fields(res_set); i++)
{
        field = mysql_fetch_field_direct(res_set, i);
        printf("column %u: name = %s max_length = %lu\n",
                i, field->name, field->max_length);
}
```

Funcția mysql_fetch_field_direct() nu funcționează în mod corespunzător anterior variantei MySQL 3.23.

- unsigned long *

mysql_fetch_lengths(MYSQL_RES *res_set);

Returnează un pointer la un tablou de valori unsigned long care reprezintă lungimile coloanelor din rândul curent al setului de rezultate. Trebuie să apelați funcția mysql_fetch_lengths() de fiecare dată când apelați mysql_fetch_row(); în caz contrar, lungimile vor fi nesincronizate cu valorile datelor.

Lungimea valorilor NULL este zero, dar o lungime zero nu indică prin prezența sa o valoare NULL. Deoarece un șir vid are de asemenea o lungime zero, trebuie să verificați dacă valoarea de date este un pointer NULL, pentru a face diferența între cele două cazuri.

Exemplul următor afișează lungimile și valorile pentru rândul curent, afișând NULL dacă valoarea este NULL:

```
unsigned long *length;

length = mysql_fetch_lengths(res_set);
for(i = 0; i < mysql_num_fields(res_set); i++)
{
        printf("lungimea este %lu, valoarea este %s\n",
                length[i], (row[i] != NULL ? row[i] : "NULL"));
}
```

Funcția mysql_fetch_lengths() a apărut în versiunea MySQL 3.20.5. Anterior versiunii MySQL 3.22.7, tipul valorii returnate de funcția mysql_fetch_lengths() er unsigned int.

- MYSQL_ROW

mysql_fetch_row(MYSQL_RES *res_set);

Returnează un pointer la următorul rând al setului de rezultate, reprezentat sub form unui tablou de șiruri (cu excepția faptului că valorile NULL din coloane sunt reprezentate sub formă de pointeri NULL). A i-a valoare din rând este al i-lea membru a tabloului de valori.

Valorile pentru toate tipurile de date, chiar și pentru tipurile numerice, sunt returnat sub formă de șiruri. Dacă doriți să efectuați un calcul numeric cu o valoare, trebuie s convertiți personal valoarea, folosind, de exemplu, funcțiile atoi() sau atof().

Funcția mysql_fetch_row() returnează NULL atunci când nu mai există rânduri în setu de rezultate. (Dacă folosiți mysql_use_result() pentru a iniția o regăsire rând cu rân a setului de rezultate, mysql_fetch_row() va returna de asemenea NULL dacă se pro duce o eroare de comunicații.)

Universitatea Tehnică Al BIBLIOTECA

Valorile de date se termină cu nul, dar nu tratați valorile care pot conține date binare drept șiruri terminate cu nul. Acestea trebuie tratate ca șiruri numerotate. (Pentru a determina lungimile valorilor din coloane, apelați funcția `mysql_fetch_lengths()`.)

Programul următor prezintă modul de parcurgere ciclică a unui rând cu valori de date și de a determina dacă fiecare valoare este sau nu `NULL`:

```
MYSQL_ROW           row;
unsigned int        i;

while ((row = mysql_fetch_row(res_set)) != NULL)
{
        for (i = 0; i < mysql_num_fields(res_set); i++)
        {
                printf("coloana %u: valoarea este %s\n",
                        i, (row[i] = = NULL ? "NULL" : "not NULL"));
        }
}
```

Pentru a determina tipurile valorilor din coloane, folosiți metadatele despre coloane stocate în structurile `MYSQL_FIELD` de informații privind coloanele, obținute prin apelarea uneia din funcțiile `mysql_fetch_field()`, `mysql_fetch_fields()` sau `mysql_fetch_field_direct()`.

- `unsigned int`

`mysql_field_count``(MYSQL *conn);`

Returnează numărul de coloane pentru cea mai recentă interogare asupra conexiunii date. Această funcție este folosită de obicei când funcția `mysql_store_result()` sau `mysql_use_result()` returnează `NULL`. Funcția `mysql_field_count()` arată dacă ar fi fost necesară sau nu returnarea unui set de rezultate. O valoare returnată egală cu 0 indică absența unui set de rezultate și absența erorilor. O valoare diferită de zero arată că funcția aștepta coloane și că, deoarece nu a fost returnată nici o coloană, s-a produs o eroare.

Exemplul următor ilustrează modul de utilizare a funcției `mysql_field_count()` pentru detecția erorilor:

```
res_set = mysql_store_result (conn);
if (res_set == NULL)  /* nu a fost returnat nici un set de rezultate */
{
        /*
         * lipsa unui set de rezultate inseamna ca s-a produs o
         * eroare sau ca nu a fost returnat nici un set de rezultate?
         */
        if (mysql_field_count (conn) > 0)
        {
                /*
                 * se astepta un set de rezultate, dar
                 * mysql_store_result() nu a returnat nimic;
                 * aceasta inseamna ca s-a produs o eroare
```



```
                 */
                print_error (conn,
                  "Problema la prelucrarea setului de rezultate");
        }
        else
        {
                /*
                 * nu a fost returnat nici un set de
                 * rezultate; interogarea nu a returnat date
                 * (nu a fost de tip SELECT, SHOW, DESCRIBE
                 * sau EXPLAIN), deci raporteaza numarul de
                 * randuri afectate de interogare
                 */
                printf ("%lu randuri afectate\n",
                  (unsigned long) mysql_affected_rows (conn));
        }
}
else /* a fost returnat un set de rezultate */
{
        /* prelucreaza randurile aici */
        mysql_free_result (res_set);
}
```

Funcția `mysql_field_count()` a fost introdusă în versiunea MySQL 3.22.24. Anterior acestei versiuni, funcția `mysql_nume_fields()` era folosită în același scop. Pentru a scrie programe care să funcționeze cu orice versiune a sistemului MySQL, includeți următorul fragment în orice fișier care folosește funcția `mysql_field_count()`:

```
#if !defined(MYSQL_VERSION_ID) || MYSQL_VERSION_ID<32224
#define mysql_field_count mysql_num_fields
#endif
```

Această secvență de instrucțiuni mapează funcția `mysql_field_count()` pe funcția `mysql_num_fields()`atunci când codul sursă este compilat sub versiuni mai vechi ale programului MySQL.

- `MYSQL_FIELD_OFFSET`

`mysql_field_seek``(MYSQL_RES *res_set, MYSQL_FIELD_OFFSET offset);`

Caută structura cu informații despre coloane specificată de `offset`. Următorul apel la funcția `mysql_fetch_field()` va returna structura cu informații pentru coloana dată. `offset` *nu* este un index de coloană; este o valoare `MYSQL_FIELD_OFFSET` obținută printr-un apel anterior la una din funcțiile `mysql_field_tell()` sau `mysql_field_seek()`.

Pentru a reinițializa la prima coloană, transferați o valoare `offset` egală cu 0.

- `MYSQL_FIELD_OFFSET`

`mysql_field_tell``(MYSQL_RES *res_set);`

Returnează offset-ul structurii curente cu informații privind coloanele. Această valoare poate fi transferată funcției `mysql_field_seek()`.

- void

mysql_free_result(MYSQL_RES *res_set);

Anulează alocarea memoriei folosite pentru setul de rezultate. Trebuie să apelați funcția mysql_free_result() pentru fiecare set de rezultate pe care îl utilizați. În mod caracteristic, seturile de rezultate sunt generate prin apelarea funcției mysql_store_result() sau a funcției mysql_use_result(). Totuși, unele funcții din biblioteca client generează seturi de rezultate în mod implicit, iar dumneavoastră sunteți responsabil și cu eliberarea memoriei alocate acestor seturi. Aceste funcții sunt mysql_list_dbs(), mysql_list_fields(), mysql_list_processes() și mysql_list_tables().

- char *

mysql_info (MYSQL *conn);

Returnează un șir care conține informații despre efectul celei mai recent executate interogări având unul din următoarele tipuri. Formatul șirului este dat imediat după fiecare interogare:

```
ALTER TABLE ...
    Records: 0 Duplicates: 0 Warnings: 0
INSERT INTO ... SELECT ...
    Records: 0 Duplicates: 0 Warnings: 0
INSERT INTO ... VALUES (...),(...),...
    Records: 0 Duplicates: 0 Warnings: 0
LOAD DATA ...
    Records: 0 Duplicates: 0 Warnings: 0
UPDATE ...
    Records: 0 Duplicates: 0 Warnings: 0
```

Valorile numerice vor varia în funcție de interogarea pe care ați executat-o, desigur.

Pentru instrucțiunile care nu au fost afișate în lista precedentă, funcția mysql_info() returnează NULL. Funcția mysql_info() returnează valori diferite de NULL pentru interogarea INSERT INTO ... VALUES numai dacă instrucțiunea conține mai mult de o listă de valori.

Șirul returnat de funcția msyql_info() folosește limbajul utilizat de server, deci nu vă puteți baza pe posibilitatea de a-l analiza prin căutarea anumitor cuvinte.

- my_ulonglong

mysql_insert_id(MYSQL conn);

Returnează valoarea AUTO_INCREMENT generată de interogarea cel mai recent executată pe conexiunea dată. Returnează zero dacă nu a fost executată nici o interogare sau dacă interogarea anterioară nu a generat o valoare AUTO_INCREMENT. Aceasta înseamnă că trebuie să apelați funcția mysql_insert_id() imediat după o interogare la care vă așteptați să genereze o valoare nouă. Dacă o altă interogare intervine între interogarea respectivă și locul în care doriți să utilizați valoarea, valoarea funcției mysql_insert_id() va fi reinițializată de interogarea intermediară.

O valoare returnată egală cu zero este diferită de orice valoare AUTO_INCREMENT, deoarece aceste valori încep de la 1. (Excepție: dacă creați o coloană AUTO_INCREMENT și apoi inserați un număr literal negativ, secvența va începe de la numărul respectiv și poate, într-un final, ajunge la 0 ca membru valabil al secvenței. În acest caz, se presupune că știți ce faceți, mai ales deoarece funcția mysql_insert_id() returnează un număr fără semn și va trebui să „jonglați" cu valoarea returnată.)

Rețineți că funcția mysql_insert_id() se comportă într-un mod diferit de funcția SQL LAST_INSERT_ID(). Funcția mysql_insert_id() este menținută de client și este stabilită pentru fiecare interogare în parte. Valoarea funcției LAST_INSERT_ID() este menținută în server și persistă de la o interogare la alta.

Valorile returnate de funcția mysql_insert_id() sunt specifice conexiunii și nu sunt afectate de activitatea AUTO_INCREMENT pe alte conexiuni.

Vezi nota despre afișarea valorilor de tipul my_ulonglong în secțiunea „Tipuri de date scalare".

- unsigned int

mysql_num_fields(MYSQL_RES *res_set);

Returnează numărul coloanelor din setul de rezultate. Funcția mysql_num_rows()este frecvent folosită pentru a parcurge ciclic coloanele rândului curent al setului, așa cum se poate vedea în următorul exemplu:

```
MYSQL_ROW     row;
unsigned int  i;

while((row = mysql_fetch_row(res_set)) != NULL)
{
        for (i = 0; i < mysql_num_fields(res_set); i++)
        {
                /* aici se executa operatii cu randul row[i]... */
        }
}
```

Anterior versiunii MySQL 3.22.24, funcția mysql_num_fields() era de asemenea folosită pentru executarea operației care este acum efectuată de mysql_field_count(), în speță verificarea dacă o valoare NULL returnată de funcția mysql_store_result() sau mysql_use_result() indică sau nu apariția unei erori. Acesta este motivul pentru care, în programele sursă mai vechi, veți vedea uneori că funcția mysql_num_fields() este apelată cu un pointer spre o variabilă de tratare a conexiunii, în loc de un pointer la un set de rezultate. Funcția mysql_num_fields() se putea apela în ambele moduri. Utilizarea funcției mysql_num_fields() cu o variabilă de tratare a conexiunilor nu se mai practică în prezent. Pentru aceasta, trebuie să scrieți programe care folosesc funcția mysql_field_count(); descrierea funcției respective vă arată cum să o folosiți chiar și cu versiuni mai vechi ale sistemului MySQL.

- my_ulonglong

mysql_num_rows(MYSQL_RES *res_set);

Returnează numărul de rânduri ale setului de rezultate. Dacă generați setul de rezultate cu funcția mysql_store_result(), puteți apela funcția mysql_num_rows() în orice moment după aceea:

```
if ((res_set = mysql_store_result(conn) == NULL)
{
        /* acum se poate apela functia mysql_num_rows() */
}
```

Dacă generați setul de rezultate cu funcția `mysql_use_result()`, funcția `mysql_num_rows()` nu va returna valoarea corectă decât după ce ați preluat toate rândurile:

```
if ((res_set = mysql_use_result(conn)) == NULL)
{
        /* functia mysql_num_rows() nu poate fi inca apelata */
        while ((row = mysql_fetch_row(res_set)) != NULL)
        {
                /* mysql_num_rows() tot nu poate fi apelata */
        }
        /* Acum poate fi apelata functia mysql_num_rows() */
```

Vezi nota despre afișarea valorilor de tipul `my_ulonglong` în secțiunea „Tipuri de date scalare".

- `MYSQL_ROW_OFFSET`

  `mysql_row_seek``(MYSQL_RES *res_set, MYSQL_ROW_OFFSET offset);`

  Caută într-un anumit rând din setul de rezultate. Funcția `mysql_row_seek()` este similară cu `mysql_data_seek()`, dar valoarea `offset` nu este un număr de rând; `offset` trebuie să fie o valoare obținută în urma unui apel la funcția `mysql_row_tell()` sau la funcția `mysql_row_seek()`.

  Funcția `mysql_row_seek()` returnează offset-ul rândului precedent.

  Funcția `mysql_row_seek()` impune regăsirea întregului set de rezultate, deci o puteți folosi numai dacă setul de rezultate a fost creat de funcția `mysql_store_result()`, nu de funcția `mysql_use_result()`.

- `MYSQL_ROW_OFFSET`

  `mysql_row_tell``(MYSQL_RES *res_set);`

  Returnează un offset care reprezintă poziția curentă a rândului în setul de rezultate. Acesta nu este un număr de rând; valoarea poate fi transferată numai funcției `mysql_row_seek()`, nu funcției `mysql_data_seek()`.

  Funcția `mysql_row_tell()` impune regăsirea întregului set de rezultate, deci o puteți folosi numai dacă setul de rezultate a fost creat de funcția `mysql_store_result()`, nu de funcția `mysql_use_result()`.

- `MYSQL_RES *`

  `mysql_store_result``(MYSQL *conn);`

  După o interogare reușită, returnează setul de rezultate și îl stochează în client. Returnează `NULL` dacă interogarea nu returnează date sau dacă s-a produs o eroare. Când funcția `mysql_store_result()` returnează `NULL`, apelați funcția `mysql_field_count()` sau una dintre funcțiile de raportare a erorilor, pentru a determina dacă nu era de așteptat un set de rezultate sau dacă s-a produs o eroare.

  Când ați terminat lucrul cu setul de rezultate, transferați-l funcției `mysql_free_result()` pentru anularea alocării memoriei aferente acestuia.

  Vezi comparația între funcțiile `mysql_store_result()` și `mysql_use_result()` în tabelul F.7.

- `MYSQL_RES *`

  `mysql_use_result``(MYSQL *conn);`

  După o interogare reușită, inițiază o regăsire a unui set de rezultate, dar nu regăsește rânduri de date. Trebuie să apelați funcția `mysql_fetch_row()` pentru a regăsi rândurile unul câte unul. Returnează `NULL` dacă interogarea nu returnează date sau dacă s-a produs o eroare. Când funcția `mysql_use_result()` returnează `NULL`, apelați funcția `mysql_field_count()` sau una dintre funcțiile de raportare a erorilor, pentru a determina dacă nu era de așteptat un set de rezultate sau dacă s-a produs o eroare.

  Când ați terminat lucrul cu setul de rezultate, transferați-l funcției `mysql_free_result()` pentru anularea alocării memoriei aferente acestuia.

  Vezi comparația între funcțiile `mysql_store_result()` și `mysql_use_result()` în tabelul F.7.

  Atât funcția `mysql_store_result()`, cât și funcția `mysql_use_result()` sunt folosite pentru regăsirea seturilor de rezultate, dar influențează modul de utilizare a altor funcții de manipulare a setului de rezultate.

Rutine de informații

Aceste funcții furnizează informații despre client, server, versiunea protocolului și conexiunea curentă.

- `char *`

  `mysql_get_client_info``(void);`

  Returnează un șir terminat cu nul, care descrie versiunea bibliotecii client; de exemplu `"3.22.25"`.

- `char *`

  `mysql_get_host_info``(MYSQL *conn);`

  Returnează un șir terminat cu nul, care descrie conexiunea curentă; de exemplu, `"Localhost via UNIX socket"` (gazdă locală prin soclu UNIX) sau `"gazda.ta.com via TCP/IP"`.

- `unsigned int`

  `mysql_get_proto_info``(MYSQL *conn);`

  Returnează un număr care indică versiunea protocolului client-server folosită pentru conexiunea curentă.

- `char *`

  `mysql_get_server_info``(MYSQL *conn);`

  Returnează un șir terminat cu nul care descrie versiunea serverului; de exemplu, `"3.22.25-log"`. Sufixele pe care le puteți vedea după numărul versiunii serverului sunt

-log (caracteristica de consemnare în jurnal este activată), -debug (serverul rulează în mod depanare) sau -demo (serverul rulează în mod demo).

- char *

  mysql_stat(MYSQL *conn);

  Returnează un șir terminat cu nul care conține informații de stare a serverului, respectiv NULL dacă s-a produs o eroare. Formatul șirului poate suferi modificări. În prezent, este următorul:

  ```
  Uptime: 864034 Threads: 1 Questions: 32736 Slow queries: 50
  Opens: 1428 Flush tables: 1 Open tables: 61
  ```

  Aceste valori pot fi interpretate astfel:

  - Uptime este intervalul de timp (în secunde) de la pornirea serverului.
  - Threads este numărul de fire care rulează în server la un moment dat.
  - Questions este numărul de interogări pe care le-a executat serverul.
  - Slow queries este numărul de interogări care durează mai mult decât parametrul long_query_time al serverului.
  - Opens este numărul de tabele pe care le-a deschis serverul.
  - Flush tables este numărul de comenzi FLUSH, REFRESH și RELOAD executate de server.
  - Open tables este numărul de tabele deschise de server la un moment dat.

  Informațiile returnate de funcția mysql_stat() sunt similare celor raportate de comanda mysqladmin status. (De unde credeți că-și ia mysqladmin informațiile?)

- unsigned long

  mysql_thread_id(MYSQL *conn);

  Returnează numărul firului pe care serverul îl asociază conexiunii curente. Puteți folosi acest număr ca identificator pentru funcția mysql_kill().

  Se recomandă să executați funcția mysql_thread_id() chiar înainte de a avea nevoie de valoarea respectivă. Dacă obțineți valoarea și o stocați, intenționând să o folosiți mai târziu, valoarea poate fi incorectă. Aceasta se poate întâmpla în cazul în care conexiunea dumneavoastră se întrerupe și este apoi restabilită (de exemplu cu ajutorul funcției mysql_ping()), deoarece serverul va atribui un nou identificator de fir.

Rutine administrative

Funcțiile din această secțiune vă permit să controlați aspecte ale modului de operare a serverului.

- int

  mysql_kill(MYSQL *conn, unsigned long thread_id);

  Suprimă firul de server identificat prin thread_id.

  Dacă aveți privilegiul PROCESS, puteți suprima orice fir. În caz contrar, vă puteți suprima numai propriile dumneavoastră fire.

Funcția mysql_kill() returnează zero în caz de succes, respectiv o valoare diferită de zero în caz de eșec.

- int

  mysql_refresh(MYSQL *conn, unsigned int options);

  Această funcție are un efect similar cu funcția SQL FLUSH, cu excepția faptului că puteți cere serverului să golească simultan mai multe categorii de entități. Valoarea options trebuie să fie una sau mai multe dintre valorile prezentate în tabelul F.8.

Tabelul F.8 Opțiuni pentru funcția mysql_refresh()

Valoarea opțiunii	Acțiune executată de server
REFRESH_GRANT	Reîncarcă informațiile conținute în tabelele de acordare
REFRESH_LOG	Începe jurnale generale și de actualizare noi (cele deschise la un moment dat)
REFRESH_TABLES	Închide toate tabelele deschise
REFRESH_HOSTS	Golește zona cache aferentă gazdelor
REFRESH_STATUS	Readuce variabilele de stare la valoarea zero

Opțiunile din tabelul F.8 sunt valori ale unor biți, deci le puteți combina folosind caracterele | sau +; de exemplu REFRESH_LOG|REFRESH_TABLES sau REFRESH_LOG+REFRESH_TABLES.

Pentru mai multe informații despre operațiile de golire, consultați descrierea instrucțiunii FLUSH din anexa D, „Referință de sintaxă SQL".

Funcția mysql_refresh() returnează zero în caz de reușită, respectiv o valoare diferită de zero în caz de eșec.

- int

  mysql_shutdown(MYSQL *conn);

  Indică serverului să-și înceteze execuția. Pentru această operație, trebuie să dispuneți de privilegiul SHUTDOWN.

  Funcția mysql_shutdown() returnează zero în caz de reușită, respectiv o valoare diferită de zero în caz de eșec.

Rutine de depanare

Aceste funcții vă permit să generați informații de depanare la extremitatea client sau server a conexiunii. Aceasta impune ca sistemul MySQL să fie compilat astfel încât să accepte depanarea. (Utilizați opțiunea --with-debug atunci când configurați distribuția MySQL.)

- void

  mysql_debug(char *debug_string);

  Execută o operație DBUG_PUSH folosind șirul debug_string. Formatul șirului este descris în manualul de referință MySQL.

  Pentru a putea folosi funcția mysql_debug(), biblioteca client trebuie compilată cu suportul pentru depanare.

- int

  mysql_dump_debug_info(MYSQL *conn);

  Indică serverului să scrie informații de depanare în jurnal. Pentru a executa această operație, trebuie să dispuneți de privilegiul PROCESS.

  Funcția mysql_dump_debug_info()returnează zero în caz de reușită, respectiv o valoare diferită de zero în caz de eșec.

Rutine scoase din uz

Biblioteca client MySQL include un număr de funcții care în prezent sunt scoase din uz, deoarece există modalități preferate de a executa aceleași operații. Majoritatea acestor funcții pot fi înlocuite prin transferul unei interogări echivalente către funcția mysql_query(). De exemplu, apelul mysql_create_db("nume_db") se poate înlocui cu următorul apel:

```
mysql_query(conn, "CREATE DATABASE nume_db")
```

Câteva funcții, printre care mysql_connect() și mysql_eof() sunt scoase din uz deoarece au fost înlocuite de funcții care execută mai multe operații sau care furnizează o cantitate mai mare de informații.

În timp, tot mai multe funcții sunt scoase din uz, deoarece MySQL înțelege tot mai multe instrucțiuni SQL. De exemplu, când a fost adăugată instrucțiunea SQL FLUSH PRIVILEGES, funcția mysql_reload() a devenit depășită. Descrierile următoare indică versiunea sistemului MySQL când funcția respectivă a fost scoasă din uz, respectiv modalitatea preferată de execuție a fiecărei funcții în prezent. Dacă biblioteca dumneavoastră client este mai veche decât versiunea MySQL menționată, trebuie să folosiți funcția scoasă din uz, desigur.

Dacă doriți să planificați aplicații viitoare, trebuie să evitați toate funcțiile enumerate în această secțiune. Unele sau chiar toate vor dispărea în versiunea MySQL 4.0.

- MYSQL

  mysql_connect(MYSQL *conn, char *nume_gazda,
  char *nume_utilizator, char *parola);

  Aceasta este predecesoarea funcției mysql_real_connect(). De fapt, este în prezent implementată ca un apel la funcția mysql_real_connect().

  Această funcție este scoasă din uz începând cu versiunea MySQL 3.22.0.

- int

  mysql_create_db(MYSQL *conn, char *nume_db);

  Creează o bază de date cu numele dat de nume_db. Această operație este acum posibilă prin emiterea unei comenzi CREATE DATABASE cu funcția mysql_query().

  Funcția mysql_create_db() returnează zero în caz de reușită, respectiv o valoare diferită de zero în caz de eșec.

  Această funcție este scoasă din uz începând cu versiunea MySQL 3.21.15.

- int

  mysql_drop_db(MYSQL *conn, char *nume_db);

  Șterge baza de date specificată de nume_db. În prezent, această operație se poate executa prin emiterea unei instrucțiuni DROP DATABASE cu funcția mysql_query().

  Funcția mysql_drop_db() returnează zero în caz de reușită, respectiv o valoare diferită de zero în caz de eșec.

  Această funcție este scoasă din uz începând cu versiunea MySQL 3.21.15.

- my_bool

  mysql_eof(MYSQL_RES *res_set);

  Returnează o valoare diferită de zero dacă s-a obținut un set de rezultate, respectiv zero dacă s-a produs vreo eroare. Funcția mysql_eof() este folosită când utilizați combinația între funcțiile mysql_use_result() pentru a iniția o operație de regăsire a unui set de rezultate și mysql_fetch_row() pentru a regăsi rândurile de date unul câte unul. Cu funcția mysql_use_result(), o valoare NULL returnată de funcția mysql_fetch_row() poate însemna fie că s-a ajuns la sfârșitul setului de rezultate, fie apariția unei erori. Funcția mysql_eof() sesizează diferența între cele două rezultate.

  În prezent, pentru a obține același rezultat se pot folosi funcțiile mysql_errno() și mysql_error(), deși cele două funcții returnează o cantitate mai mare de informații. (Funcțiile indică și motivul producerii oricărei erori, nu se rezumă la a semnala pur și simplu producerea erorii.)

  Această funcție a fost scoasă din uz începând cu versiunea MySQL 3.21.17.

- MYSQL_RES *

  mysql_list_dbs(MYSQL *conn, char *wild);

  Returnează un set de rezultate care enumeră numele bazelor de date din server, respectiv NULL dacă s-a produs o eroare. Această listă conține toate bazele de date care corespund expresiei regulate SQL indicate de wild (care poate conține caracterele de înlocuire % și _) sau toate bazele de date, dacă wild este NULL. Dumneavoastră sunteți responsabil cu apelarea funcției mysql_free_result() pentru anularea alocării memoriei aferente setului de rezultate.

  Lista produsă de funcția mysql_list_dbs() poate fi obținută prin executarea unei comenzi SHOW DATABASES cu mysql_query() și apoi prin prelucrarea setului de rezultate.

  Această funcție a fost scoasă din uz începând cu versiunea MySQL 3.22.0.

- MYSQL_RES *

  mysql_list_fields(MYSQL *conn, char *nume_tbl, char *wild);

  Returnează un set de rezultate care enumeră numele coloanelor din tabelul dat, respectiv NULL dacă se produce vreo eroare. Lista conține numele tuturor coloanelor care corespund expresiei regulate SQL indicate de wild (care poate conține caracterele de înlocuire % și _) sau toate coloanele, dacă wild este NULL. Dumneavoastră sunteți responsabil cu apelarea funcției mysql_free_result() pentru anularea alocării memoriei aferente setului de rezultate.

Lista produsă de funcția `mysql_list_fields()` poate fi obținută prin executarea unei comenzi `SHOW COLUMNS` cu `mysql_query()` și apoi prin prelucrarea setului de rezultate.

Această funcție a fost scoasă din uz începând cu versiunea MySQL 3.22.0.

- `MYSQL_RES *`

  `mysql_list_processes``(MYSQL *conn);`

  Returnează un set de rezultate care conține o listă a proceselor care rulează pe server, respectiv `NULL` dacă s-a produs o eroare. Dacă dispuneți de privilegiul `PROCESS`, lista conține toate procesele server. Dacă nu, lista conține propriile dumneavoastră procese. Dumneavoastră sunteți responsabil cu apelarea funcției `mysql_free_result()` pentru anularea alocării memoriei aferente setului de rezultate.

  Lista produsă de funcția `mysql_list_fields()` poate fi obținută prin executarea unei comenzi `SHOW PROCESSLIST` cu `mysql_query()` și apoi prin prelucrarea setului de rezultate.

  Această funcție a fost scoasă din uz începând cu versiunea MySQL 3.22.0.

- `MYSQL_RES *`

  `mysql_list_tables``(MYSQL *conn, char *wild);`

  Returnează un set de rezultate care enumeră numele tabelelor din baza de date curentă, respectiv `NULL` dacă se produce vreo eroare. Lista conține numele tuturor coloanelor care corespund expresiei regulate SQL indicate de `wild` (care poate conține caracterele de înlocuire `%` și `_`) sau toate tabelele, dacă `wild` este `NULL`. Dumneavoastră sunteți responsabil cu apelarea funcției `mysql_free_result()` pentru anularea alocării memoriei aferente setului de rezultate.

  Lista produsă de funcția `mysql_list_tables()` poate fi obținută prin executarea unei comenzi `SHOW TABLES` cu `mysql_query()` și apoi prin prelucrarea setului de rezultate.

  Această funcție a fost scoasă din uz începând cu versiunea MySQL 3.22.0.

- `int`

  `mysql_reload``(MYSQL *conn);`

  Indică serverului să reîncarce tabelele de acordare. Această operație se poate executa în prezent prin emiterea unei interogări `FLUSH PRIVILEGES` cu funcția `mysql_query()`. Pentru a folosi funcția `mysql_reload()`, trebuie să dispuneți de privilegiul `RELOAD`.

  Funcția `mysql_reload()` returnează zero în caz de reușită, respectiv o valoare diferită de zero în caz de eșec.

  Această funcție a fost scoasă din uz începând cu versiunea MySQL 3.21.9.

ANEXA G

Referință API Perl DBI

Această anexă descrie interfața de programare a aplicațiilor Perl DBI. Interfața constă dintr-un set de metode și atribute pentru comunicarea cu serverele de baze de date și pentru accesul la bazele de date din scripturile Perl. De asemenea, anexa descrie extensiile la DBI specifice sistemului MySQL furnizate de DBD::mysql, driverul MySQL pentru baze de date.

Unele metode și atribute DBI nu vor fi discutate aici, fie deoarece nu se aplică sistemului MySQL, fie deoarece sunt metode noi, experimentale, care se pot modifica în timpul dezvoltării sau care au fost chiar abandonate. Unele metode DBD specifice sistemului MySQL nu au fost discutate, deoarece sunt depășite. Dacă doriți mai multe informații despre metodele noi sau despre cele depășite, consultați documentația DBI sau documentația driverului DBD pentru MySQL, pe care o puteți obține prin rularea următoarelor comenzi:

```
% perldoc DBI
% perldoc DBI::FAQ
% perldoc DBD::mysql
```

Această anexă servește drept referință, deci include numai scurte fragmente de program care ilustrează utilizarea interfeței API pentru Perl DBI. Pentru scripturi client complete și note privind redactarea lor, consultați capitolul 7, „Interfața API pentru Perl DBI".

Scrierea scripturilor

Fiecare script Perl care folosește modulul DBI trebuie să includă următoarea linie:

```
use DBI;
```

Nu este necesară includerea unei linii `use` pentru un anumit modul de nivel DBD, deoarece DBI va activa modulul adecvat atunci când vă conectați la server.

În mod normal, un script DBI va deschide o conexiune folosind metoda `connect()`, va emite interogări și apoi va închide conexiunea cu metoda `disconnect()`. Metodele folosite pentru emiterea interogărilor variază. De regulă, interogările non-`SELECT` sunt executate prin transferul interogării către funcția `prepare()`, apoi apelarea metodei `execute()` și în final regăsirea rând cu rând a rezultatelor interogării într-un ciclu care invocă în mod repetat o metodă de preluare a rândurilor, precum `fetchrow_array()` sau `fetchrow_hashref()`.

Metode DBI

Descrierile metodelor prezentate aici sunt scrise într-un format oarecum diferit de cel folosit pentru funcțiile C din Anexa F, „Referință API C", respectiv de formatul funcțiilor PHP care va fi prezentat în Anexa H, „Referință API PHP". Funcțiile din aceste anexe sunt scrise în formă de prototip, enumerând în mod explicit tipurile valorilor returnate și tipurile de parametri. În general, Perl nu folosește prototipuri în acest mod, deci descrierile din această anexă indică tipurile valorilor returnate și tipurile parametrilor ca variabile, unde caracterul inițial al variabilei indică tipul valorii: `$` pentru un scalar, `@` pentru un tablou și `%` pentru o valoare hash (tablou asociativ). În plus, pentru orice parametru care începe cu un caracter `\`, este necesar transferul unei referințe la variabila respectivă, nu al variabilei în sine. Un nume de variabilă care se termină cu particula `_ref` arată că valoarea variabilei este o referință.

În această anexă se vor folosi numeroase variabile cu semnificații specifice. Aceste variabile sunt prezentate în tabelul G.1.

Tabelul G.1 Nume convenționale de variabile Perl

Nume	Semnificație
`$drh`	Variabilă de manipulare a unui obiect driver
`$dbh`	Variabilă de manipulare a unui obiect bază de date
`$sth`	Variabilă de manipulare a unui obiect instrucțiune (interogare)
`$fh`	Variabilă de manipulare a unui fișier deschis
`$h`	Variabilă de manipulare „generală"; semnificația depinde de conținut
`$rc`	Codul returnat de operații care returnează „adevărat" sau „fals"
`$rv`	Valoarea returnată de operații care returnează un întreg
`$rows`	Valoarea returnată de operații care returnează un număr de rânduri
`@ary`	Un tablou (listă) care reprezintă un rând de valori returnate de o interogare

Multe metode acceptă un argument hash `%attr`, care conține atribute ce influențează modul de funcționare a metodei. Acest argument hash trebuie transferat prin referință. Puteți face aceasta în două moduri. O modalitate este stabilirea conținutului valorii hash `%attr` înainte de a invoca metoda, iar apoi transferul acesteia către metodă:

```
my (%attr) = (AttrName1 => valoare1, AttrName2 => valoare2);
$ret_val = $h->method(..., \%attr);
```

Cealaltă metodă constă în furnizarea unui hash anonim direct în declarația de invocare a metodei:

```
$ret_val = $h->method
    (...,{AttrName1 => valoare1, AttrName2 => valoare2});
```

Informațiile opționale sunt indicate prin paranteze drepte ([ ]). Modul de utilizare a unei metode sau a unei funcții este indicat de către secvența apelantă. `DBI-` indică o metodă de clasă DBI, `DBI::` indică o funcție DBI, iar `$DBI::` indică o variabilă DBI. Pentru metodele care sunt apelate folosind variabile de manipulare, numele variabilei de manipulare indică aria de acțiune a metodei. `$dbh-` indică o metodă de manipulare a unei baze de date, `$sth-` indică o metodă de manipulare a instrucțiunilor, iar `$h-` indică o metodă care poate fi apelată cu diferite tipuri de variabile de manipulare. Iată un exemplu de secvență apelantă:

```
@row_ary = $dbh->selectrow_array ($instructiune,
            [\%attr [, @valori_asociere]]);
```

Această secvență arată că metoda `selectrow_array()` a fost apelată ca metodă de manipulare a bazelor de date, deoarece este invocată folosind `$dbh->`. Parametrii sunt `$instructiune` (o valoare scalară), `%attr` (o valoare hash care trebuie transferată ca referință, așa cum o arată caracterul `\` de la început) și `@valori_asociere` (un tablou). Al doilea și al treilea parametru sunt opționali. Valoarea returnată este un tablou.

Descrierile metodelor indică valoarea returnată în caz de eroare, dar o eroare poate fi returnată numai dacă atributul `RaiseError` este dezactivat. Dacă acest atribut este activat, scriptul își va încheia execuția în mod automat.

În descrierile care urmează, termenul „interogare `SELECT`" va desemna atât o interogare `SELECT`, cât și orice altă interogare care returnează rânduri, cum ar fi `DESCRIBE`, `EXPLAIN` sau `SHOW`.

Metodele de clasă DBI

Parametrul `%attr` pentru metodele din această secțiune se poate folosi pentru a specifica atribute de prelucrare a metodelor. Pentru MySQL, cele mai importante atribute sunt `PrintError` și `RaiseError`. De exemplu, pentru a activa terminarea automată a execuției scriptului la apariția unei erori DBI, activați atributul `RaiseError`:

```
$dbh = DBI->connect($sursa_date, $nume_utilizator, $parola,
    {RaiseError => 1});
```

Despre atributele `PrintError` și `RaiseError` vom discuta în secțiunea „Atribute generale ale variabilelor manipulare".

- `@ary = DBI->available_drivers([$quiet]);`

  Returnează o listă a driverelor DBI disponibile. Dacă sunt găsite mai multe drivere cu același nume, va fi emis un mesaj de avertizare. Puteți suprima acest mesaj transferând o valoare egală cu 1 a parametrului `$quiet`.

- `$dbh = DBI->connect($sursa_date, $nume_utilizator, $parola [, \%attr]);`

  Metoda `connect()` stabilește o conexiune cu serverul de baze de date și returnează o variabilă de manipulare pentru baze de date. Variabila este `undef` dacă încercarea de conexiune eșuează. Pentru a termina o conexiune stabilită cu succes, invocați metoda `disconnect()` folosind variabila de manipulare pentru baze de date returnată de metoda `connect()`.

  Pentru metoda `connect()`, toate atributele specificate în parametrul `%attr` influențează global prelucrarea metodelor DBI.

  Sursa de date poate fi dată în mai multe forme. Prima parte este întotdeauna `DBI::mysql`, unde `DBI` poate fi scris atât cu majuscule, cât și cu minuscule, iar numele driverului, în speță `mysql`, trebuie scris cu minuscule. Toate informațiile scrise în sursa

de date după numele driverului sunt interpretate de driver, deci sintaxa descrisă în expunerea următoare nu se aplică în mod obligatoriu oricărui alt driver decât DBD::mysql.

După numele driverului, mai puteți specifica un nume al bazei de date și un nume de gazdă în partea inițială a șirului care reprezintă sursa de date:

```
$sursa_date = "DBI:mysql:nume_db";
$sursa_date = "DBI:mysql:nume_db:nume_gazda";
```

Baza de date poate fi specificată sub forma *nume_db* sau *database=nume_db*. Numele gazdei poate fi specificat sub forma *nume_gazda* sau *host=nume_gazda*.

După partea de început a șirului care reprezintă sursa de date, puteți specifica mai multe opțiuni folosind formatul *atribut=valoare*. Fiecare opțiune trebuie precedată printr-un caracter punct și virgulă. Driverul MySQL acceptă următoarele opțiuni:

- `host=nume_gazda`

  Gazda la care se va stabili conexiunea. Se mai poate specifica și un număr de port, folosind formatul *nume_gazda:numar_port*.

- `port=numar_port`

  Numărul portului la care se va stabili conexiunea.

- `mysql_compression = 1`

  Această opțiune activează protocolul client-server de comunicare comprimată între client și serverul MySQL.

  Opțiunea `mysql_compression` necesită versiunea MySQL 3.22.3 sau o versiune ulterioară, respectiv versiunea DBD::mysql 1.19.20 sau o versiune ulterioară.

- `mysql_read_default_file=nume_fisier`

  În mod prestabilit, scripturile DBI nu citesc fișiere cu opțiuni MySQL pentru a obține parametrii de conexiune. Această opțiune vă permite să specificați un fișier de opțiune care va fi citit. Numele fișierului trebuie să reprezinte calea completă de acces la acesta. (În caz contrar, va fi interpretat relativ la catalogul unde executați scriptul, iar dumneavoastră veți obține rezultate inconsecvente.)

  Pentru a citi fișierul `.my.cnf` din catalogul de bază al oricărui utilizator care se întâmplă să ruleze scriptul, specificați numele fișierului sub forma `$ENV{HOME}/.my.cnf`. Această notație este utilă când un script va fi utilizat de mai multe persoane și dumneavoastră doriți ca acestea să se conecteze sub propriile conturi, nu sub contul unui utilizator pe care îl codați hard în script.

  Opțiunea `mysql_read_default_file` necesită MySQL versiunea 3.22.10 sau o versiune ulterioară, respectiv versiunea DBD::mysql 1.21.06 sau o versiune ulterioară.

- `mysql_read_default_group=nume_grup`

  În cazul în care citiți un fișier cu opțiuni pentru a determina parametrii de conexiune, grupul `[client]` este citit în mod prestabilit. Opțiunea `mysql_read_default_group` vă permite să specificați un grup care va fi citit în afară de grupul `[client]`. De exemplu, `mysql_read_default_group=dbi` specifică faptul că, în afara grupului `[client]`, mai trebuie citit și grupul `[dbi]`. Precedența parametrilor care se găsesc în mai multe grupuri este descrisă în anexa E, „Referință de programe MySQL".

  Opțiunea `mysql_read_default_group` necesită MySQL versiunea 3.22.10 sau o versiune ulterioară, respectiv versiunea DBD::mysql 1.21.06 sau o versiune ulterioară.

- `mysql_socket=nume_soclu`

  Această opțiune specifică numele căii de acces la soclul de domeniu UNIX care va fi utilizat pentru conexiunile cu `localhost`, în cazul în care calea spre soclul local nu este convenabilă.

  Opțiunea `mysql_socket` necesită MySQL 3.21.15 sau o versiune ulterioară. DBI consultă mai multe variabile de mediu pentru a determina parametrii de conexiune:

  - Dacă sursa de date este nedefinită sau vidă, se va folosi valoarea variabilei `DBI_DSN`.
  - Dacă numele driverului lipsește din sursa de date, se va folosi valoarea variabilei `DBI_DRIVER`.
  - Dacă parametrii `nume_utilizator` sau `parola` ai apelului la metoda `connect()` nu sunt definiți, se vor folosi valorile variabilelor `DBI_USER` și `DBI_PASS`. Acest fapt *nu* se întâmplă dacă parametrii sunt alcătuiți din șiruri vide.

  Utilizarea variabilei `DBI_PASS` reprezintă un pericol la adresa securității, deci nu trebuie să o folosiți în nici o situație în care securitatea este importantă. (Valorile variabilelor de mediu sunt vizibile altor utilizatori prin intermediul unor comenzi precum `ps`.)

  Dacă unii parametri de conexiune rămân necunoscuți după consultarea tuturor surselor de informații, DBI folosește valorile prestabilite. Dacă numele gazdei nu este specificat, se va folosi numele prestabilit `localhost`. Dacă numele utilizatorului nu este specificat, acesta va avea ca valoare prestabilită numele dumneavoastră de deschidere a sesiunii de lucru (sub UNIX), respectiv `ODBC` (sub Windows). Dacă nu este specificată parola, nu există nici o valoare prestabilită. Nu va fi trimisă nici o parolă.

- `$drh = DBI->install_driver($driver_name)`

  Activează un driver de nivel DBD și returnează o variabilă de manipulare a acestuia. Pentru MySQL, numele driverului este `"mysql"` (trebuie scris cu minuscule). În mod normal, utilizarea acestei metode nu este necesară, deoarece DBI activează în mod automat driverul adecvat atunci când invocați metoda `connect()`. Totuși, metoda `install_driver()` poate fi utilă dacă folosiți metoda `func()` pentru a efectua operații administrative. (Vezi secțiunea „Metode administrative specifice sistemului MySQL".)

Metode de manipulare a bazelor de date

Metodele din această secțiune sunt invocate cu ajutorul unei variabile de manipulare a bazelor de date și se pot folosi după ce ați obținut o atare variabilă prin apelarea metodei `connect()`.

Parametrul `%attr` pentru metodele din această secțiune se poate folosi pentru a specifica atribute de prelucrare a metodelor. Pentru MySQL, cele mai importante din acestea sunt `PrintError` și `RaiseError`. De exemplu, puteți activa `RaiseError` pentru a determina încheierea automată a execuției scriptului în cazul producerii unei erori DBI în timpul prelucrării unei anumite interogări, ca aceasta:

```
$rows = $dbh-do($instructiune, {Raise_Error = 1});
```

Despre atributele PrintError și RaiseError vom discuta în secțiunea „Atribute generale de manipulare".

- `$rc = $dbh->disconnect();`

  Termină conexiunea asociată variabilei de manipulare. Dacă o conexiune rămâne activă când un script își încheie execuția, este afișat un mesaj de avertizare și conexiunea este terminată automat.

- `$rows = $dbh->do($instructiune [, \%attr [, @valori_asociere]]);`

  Pregătește și execută interogarea indicată de $instructiune. Valoarea returnată reprezintă numărul de rânduri afectate, respectiv –1 dacă numărul de rânduri este necunoscut sau undef dacă se produce o eroare. Dacă numărul de rânduri afectate este zero, valoarea returnată este șirul 0E0, care este echivalent cu zero, dar este considerat adevărat.

  Metoda do() este folosită mai ales pentru instrucțiuni care nu regăsesc rânduri, precum DELETE, INSERT, REPLACE și UPDATE. Dacă încercați să folosiți această metodă pentru o instrucțiune SELECT, nu veți primi o variabilă de manipulare a instrucțiunilor și nu veți putea prelua rânduri.

  Parametrul @valori_asociere este folosit când instrucțiunea conține câmpuri de înlocuire (indicate prin caracterul ? plasat în șirul de interogare). @valori_asociere este o listă a valorilor care vor fi asociate câmpurilor de înlocuire. Este necesar ca numărul valorilor să fie egal cu acela al câmpurilor de înlocuire. Dacă specificați valorile care trebuie asociate, dar fără atribute, transferați undef ca valoare a parametrului \%attr.

- `$rc = $dbh->ping();`

  Verifică dacă mai este activă conexiunea cu serverul, după care returnează true sau false în consecință.

- `$sth = $dbh->prepare($instructiune [, \%attr]);`

  Pregătește interogarea indicată de $instructiune în vederea unei execuții ulterioare și returnează o variabilă de manipulare a instrucțiunii. Variabila de manipulare returnată se poate folosi cu metoda execute() pentru executarea interogării.

- `$str = $dbh->quote($valoare [, $tip_date]);`

  Prelucrează un șir pentru a efectua încadrarea între ghilimele și modificarea semnificației unor caractere speciale din instrucțiunile SQL, astfel încât șirul să nu cauzeze o eroare de sintaxă atunci când executați instrucțiunea. De exemplu, șirul "I'm happy" este returnat sub forma "'I\'m happy'" (fără ghilimelele duble). Dacă $valoare are valoarea undef, este returnată sub forma șirului "NULL" (fără semnele citării).

  De obicei, parametrul $tip_date nu este necesar, deoarece MySQL convertește în mod automat în alte tipuri valorile specificate ca șiruri în cadrul interogărilor. $tip_date poate fi specificat ca un indiciu privind tipul unei anumite valori; de exemplu, DBI::SQL_INTEGER arată că $valoare reprezintă un întreg.

  Nu folosiți metoda quote() cu valori pe care le veți insera într-o interogare folosind caractere de înlocuire; DBI încadrează automat aceste valori între ghilimele.

- `$ary_ref = $dbh->selectall_arrayref ($instructiune [, \%attr [, @valori_asociere]]);`

  Execută interogarea specificată de $instructiune și returnează rezultatul prin combinarea metodelor prepare(), execute() și fetchall_arrayref(). Returnează undef dacă se produce o eroare.

  Dacă parametrul $instructiune este o instrucțiune pregătită anterior, etapa prepare() este omisă.

  Parametrul @valori_asociere are aceeași semnificație ca pentru metoda do().

- `$ary_ref = $dbh->selectcol_arrayref ($instructiune [\%attr [, @valori_asociere]]);`

  Execută interogarea specificată în $instructiune și returnează prima coloană a rezultatului prin combinarea metodelor prepare() și execute(). Rezultatul este returnat ca o referință la un tablou care conține prima coloană din fiecare rând. Dacă se produce o eroare, metoda returnează undef.

  Dacă parametrul $instructiune este o instrucțiune pregătită anterior, etapa prepare() este omisă.

  Parametrul @valori_asociere are aceeași semnificație ca pentru metoda do().

- `$row_ary = $dbh->selectrow_array ($instructiune [, \%attr [, @valori_asociere]]);`

  Execută interogarea specificată în $instructiune și returnează primul rând al rezultatului, prin combinarea metodelor prepare(), execute() și fetchrow_array().

  Dacă parametrul $instructiune este o instrucțiune pregătită anterior, etapa prepare() este omisă.

  Când este apelată într-un context de listă, metoda selectrow_array() returnează un tablou care reprezintă valorile din rând, respectiv un tablou vid dacă se produce o eroare. În context scalar, metoda selectrow_array() returnează valoarea primului element din tablou (prima coloană a rândului). Metoda returnează undef în cazul producerii unei erori.

  Parametrul @valori_asociere are aceeași semnificație ca pentru metoda do().

Metode de manipulare a instrucțiunilor

Metodele din această secțiune sunt invocate prin intermediul unei variabile de tratare a instrucțiunilor, pe care o obțineți prin apelarea metodei prepare().

- `$rc = $sth->bind_col($numar_coloana, \$var_de_asociat);`

  Asociază o coloană dată pentru o interogare SELECT la o variabilă Perl, care trebuie transferată ca referință. $numar_coloana trebuie să se găsească în intervalul cuprins între 1 și numărul coloanelor selectate de interogare. La fiecare preluare a unui rând, variabila este actualizată automat cu valoarea din coloană.

  Metoda bind_col() trebuie apelată după prepare() și înainte de execute().

  Metoda bind_col() returnează „fals" dacă numărul coloanei nu se află în domeniul cuprins între 1 și numărul coloanelor selectate de interogare.

- `$rc = $sth->bind_columns(\$var_de_asociat1,\$var_de_asociat2, ...)`

  Asociază o listă cu variabile coloanelor returnate de o instrucțiune `SELECT` pregătită. Examinați descrierea metodei `bind_col()`.

  Metoda `bind_columns()` returnează „fals" dacă numărul de referințe nu corespunde numărului coloanelor selectate de interogare.

- `$rv = $sth->bind_param($n, $valoare [, \%attr]);`

  `$rv = $sth->bind_param($n, $valoare [, $tip_asociere]);`

  Asociază o valoare cu un câmp de înlocuire (?) dintr-o instrucțiune. Metoda trebuie apelată după `prepare()` și înainte de `execute()`.

  `$n` specifică numărul câmpului de înlocuire la care trebuie asociată valoarea `$valoare` și trebuie să fie cuprins între 1 și numărul câmpurilor de înlocuire. Pentru a asocia valoarea `NULL`, transferați `undef`.

  Parametrul `\%attr` sau `$tip_asociere` poate fi furnizat ca indiciu privind tipul valorii care va fi asociată. De exemplu, pentru a arăta că valoarea reprezintă un întreg, puteți invoca metoda `bind_param()` în oricare din următoarele moduri:

  ```
  $rv = $sth->bind_param($n, $valoare , { TYPE => DBI::SQL_INTEGER });
  $rv = $sth->bind_param($n, $valoare , DBI::SQL_INTEGER);
  ```

  În mod prestabilit, variabila va fi tratată ca `VARCHAR`, ceea ce este suficient în condiții normale, deoarece MySQL convertește valorile de tip șir din interogări în alte tipuri de date, conform necesităților.

- `$rows = $sth->dump_results([$lungmax [, $sep_linie [, $sep_camp[, $fh]]]]);`

  Preia toate rândurile de la variabila de manipulare a instrucțiunilor `$sth`, le formatează apelând funcția utilitară `DBI::neat_list()` și le afișează în variabila de manipulare a fișierelor dată. Returnează numărul rândurilor preluate.

  Valorile prestabilite pentru parametrii `$lungmax`, `$sep_linie`, `$sep_camp` și `$fh` sunt 35, `"\n"`, `", "` și `STDOUT`.

- `$rv = $sth->execute([@valori_asociere]);`

  Execută o instrucțiune pregătită. Returnează „adevărat" dacă instrucțiunea a fost executată cu succes, respectiv `undef` dacă s-a produs o eroare.

  Parametrul `@valori_asociere` are aceeași semnificație ca în cazul metodei `do()`.

- `$ary_ref = $sth->fetch();`

  `fetch()` este un alias pentru `fetchrow_arrayref()`.

- `$tbl_ary_ref = $sth->fetchall_arrayref([$slice_array_ref]);`

  `$tbl_ary_ref = $sth->fetchall_arrayref([$slice_hash_ref]);`

  Preia toate rândurile de la variabila de manipulare a instrucțiunii `$sth` și returnează o referință la un tablou care conține o referință pentru fiecare rând preluat. Semnificația fiecărei referințe din tablou depinde de argumentul pe care îl transferați. Dacă nu este dat nici un argument sau este dat un argument referință la o „felie" de tablou, fiecare element al valorii `$tbl_ary_ref` este o referință la un tablou care conține valorile unui rând al rezultatului. Cu un argument referință la o „felie" a unei valori hash, fiecare element al valorii `$tbl_ary_ref` este o referință la o valoare hash care conține valorile unui rând al rezultatului.

- `@ary = $sth->fetchrow_array();`

  Când este apelată într-un context listă, metoda `fetchrow_array()` returnează un tablou care conține valorile coloanelor din următorul rând al setului de rezultate, respectiv un tablou vid dacă nu mai există rânduri sau dacă s-a produs o eroare. Într-un context scalar, metoda `fetchrow_array()` returnează valoarea primului element al tabloului (adică prima coloană a rândului), respectiv `undef` dacă nu mai sunt rânduri sau dacă s-a produs o eroare.

  Puteți face diferența între etalarea normală a setului de rezultate și o eroare prin verificarea valorii variabilei `$sth-err()`. O valoare egală cu zero arată că ați ajuns la sfârșitul setului de rezultate, fără erori.

- `$ary_ref = $sth->fetchrow_arrayref();`

  Returnează o referință la un tablou care conține valori din coloane aferente următorului rând al setului de rezultate. Returnează `undef` dacă nu mai sunt rânduri sau dacă se produce o eroare.

  Puteți face diferența între etalarea normală a unui set de rezultate și o eroare prin verificarea metodei `$sth-err()`. O valoare egală cu zero arată că ați ajuns la sfârșitul setului de rezultate, fără erori.

- `$hash_ref = $sth->fetchrow_hashref([$nume]);`

  Returnează o referință la o valoare hash care conține valorile coloanelor aferente următorului rând al setului de rezultate. Dacă nu mai sunt rânduri sau dacă se produce o eroare, returnează `undef`. Valorile hash de index sunt numele coloanelor, iar elementele valorii hash sunt valorile coloanelor.

  Argumentul `$nume` poate fi specificat pentru a indica atributul variabilă de manipulare a instrucțiunilor care se va folosi pentru valorile cheilor hash. Valoarea sa prestabilită este `"NAME"`. Acest fapt poate cauza probleme, deoarece numele coloanelor nu sunt sensibile la diferența între majuscule și minuscule în interogări, dar cheile hash sunt. Pentru a forța scrierea cheilor hash cu minuscule sau cu majuscule, puteți specifica o valoare `$nume` egală cu `"NAME_lc"` sau `"NAME_uc"`.

  Puteți face diferența între etalarea normală a unui set de rezultate și o eroare prin verificarea valorii variabilei `$sth-err()`. O valoare egală cu zero arată că ați ajuns la sfârșitul setului de rezultate, fără erori.

- `$rc = $sth->finish();`

  Eliberează toate resursele alocate variabilei de manipulare a instrucțiunii. În mod normal, nu este necesară invocarea în mod explicit a acestei metode, dar dacă preluați numai o parte a unui set de rezultate, prin apelarea metodei `finish()` indicați DBI că ați terminat preluarea datelor.

  Apelarea metodei `finish()` poate invalida atributele instrucțiunii; cel mai bine este să obțineți accesul la acestea imediat după ce ați invocat metoda `execute()`.

- `$rv = $sth->rows();`

  Returnează numărul rândurilor afectate de instrucțiunea asociată cu `$sth`, respectiv –1 dacă s-a produs o eroare. Această metodă este folosită mai ales pentru instrucțiuni care nu returnează rânduri. Pentru instrucțiunile `SELECT`, nu trebuie să contați pe metoda `rows()`; numărați rândurile pe măsură ce le preluați.

Metode generale de manipulare

Metodele din această secțiune nu sunt specifice unor anumite tipuri de variabile de manipulare. Metodele pot fi invocate folosind variabile de manipulare pentru drivere, baze de date sau instrucțiuni.

- `$h->err();`

  Returnează codul numeric al erorii pentru cel mai recent invocată operație cu driverul. Un cod zero indică absența oricărei erori.

- `$h->errstr();`

  Returnează mesajul care conține șirul de eroare pentru cel mai recent invocată operație cu driverul. Un șir vid indică absența oricărei erori.

- `DBI->trace($trace_level [, $trace_filename]);`

  `$h->trace($trace_level [, $trace_filename]);`

  Stabilește un nivel de urmărire. Urmărirea furnizează informații despre modul de operare a modulului DBI. Nivelul de urmărire poate varia între 0 (inactiv) și 9 (cantitate maximă de informații). Urmărirea poate fi activată pentru toate operațiile DBI din cadrul unui script prin invocarea funcției `trace()` ca o metodă de clasă DBI sau pentru o variabilă de manipulare individuală:

`DBI->trace(2);`	Activează urmărirea scriptului
`$sth->trace(2);`	Activează urmărirea variabilei de manipulare

  De asemenea, urmărirea mai poate fi activată la nivel global pentru toate scripturile DBI pe care le rulați, prin configurarea variabilei de mediu `DBI_TRACE`.

  Datele de ieșire obținute prin urmărire ajung în mod prestabilit la `STDERR`. Parametrul `$trace_filename` poate fi furnizat pentru a orienta datele de ieșire spre un alt fișier. Datele de ieșire vor fi atașate la conținutul existent al fișierului, indiferent de natura acestuia.

  Fiecare apel la funcția `trace()` determină orientarea spre același fișier a *tuturor* variabilelor de manipulare urmărite. Dacă nu este denumit nici un fișier, toate datele de ieșire obținute prin urmărire ajung la `STDERR`.

- `DBI->trace_msg($str [, $min_level]);`

  `$h->trace_msg($str [, $min_level]);`

  Scrie un mesaj în datele de ieșire obținute prin urmărire, dacă variabila de manipulare este urmărită sau dacă a fost activată urmărirea la nivelul modulului DBI. Metoda `trace_msg()` poate fi apelată sub forma `DBI-trace_msg()` pentru a scrie un mesaj dacă este activată urmărirea la nivelul modulului DBI.

  Parametrul `$min_level` poate fi precizat pentru a specifica faptul că mesajul trebuie scris numai dacă nivelul de urmărire se află cel puțin la valoarea respectivă.

Metode administrative specifice sistemului MySQL

Această secțiune descrie metoda `func()` furnizată de DBI ca mijloc de acces direct la operațiile specifice driverului.

```
$rc = $drh->func('createdb',
   $nume_db, $nume_gazda, $nume_utilizator, $parola, 'admin');
$rc = $drh->func('dropdb',
   $nume_db, $nume_gazda, $nume_utilizator, $parola, 'admin');
$rc = $drh->func('shutdown',
   $nume_gazda, $nume_utilizator, $parola, 'admin');
$rc = $drh->func('reload',
   $nume_gazda, $nume_utilizator, $parola, 'admin');

$rc = $dbh->func('createdb', $nume_db, 'admin');
$rc = $dbh->func('dropdb', $nume_db, 'admin');
$rc = $dbh->func('shutdown', 'admin');
$rc = $dbh->func('reload', 'admin');
```

Metoda `func()` este accesibilă prin intermediul unei variabile de manipulare a driverului sau al unei variabile de manipulare a bazelor de date. O variabilă de manipulare a driverului nu este asociată cu o conexiune deschisă, deci, dacă obțineți acces la metoda `func()` în acest mod, trebuie să furnizați argumente pentru numele gazdei, numele utilizatorului și parolă, pentru a permite metodei să stabilească o conexiune. Dacă obțineți acces la metoda `func()`cu o variabilă de manipulare a bazelor de date, aceste argumente nu mai sunt necesare. O variabilă de manipulare a driverului poate fi obținută, dacă este necesar, după cum urmează:

```
$drh = DBI->install_driver("mysql");   # ("mysql" trebuie
                                       # scris cu minuscule)
```

- `createdb`

  Creează baza de date specificată de `$nume_db`. Pentru a executa această operație, trebuie să dispuneți de privilegiul `CREATE` pentru baza de date.

- `dropdb`

  Șterge (elimină) baza de date specificată de `$nume_db`. Pentru a executa această operație, trebuie să dispuneți de privilegiul `DROP` pentru baza de date.

  Fiți atent: dacă ștergeți o bază de date, s-a terminat cu ea. Nu o puteți aduce înapoi.

- `shutdown`

  Oprește serverul. Pentru a executa această operație, trebuie să dispuneți de privilegiul `SHUTDOWN`.

- `reload`

  Indică serverului să reîncarce tabelele de acordare. Această operație este necesară dacă modificați în mod direct conținutul tabelelor de acordare, folosind instrucțiunile

DELETE, INSERT sau UPDATE în detrimentul instrucțiunilor GRANT și REVOKE. Pentru a putea folosi comanda reload, trebuie să dispuneți de privilegiul RELOAD.

Funcții utilitare din modulul DBI

Aceste funcții sunt invocate sub forma DBI::*nume_functie*(), nu sub forma DBI->*nume_functie*().

- @bool = DBI::**looks_like_number**(@tablou);

  Preia o listă cu valori și returnează un tablou care conține un membru pentru fiecare element al listei. Fiecare membru indică dacă argumentul corespunzător arată ca un număr: „adevărat" dacă această similitudine există, „fals" în caz contrar, undef dacă argumentul este nedefinit sau vid.

- $str = DBI::**neat($valoare** [, $lungmax]);

  Returnează un șir care conține o reprezentare estetic formatată a argumentului $valoare. Șirurile sunt încadrate între ghilimele, numerele nu. (Rețineți însă că numerele încadrate între ghilimele vor fi considerate șiruri.) Valorile nedefinite sunt raportate ca undef, iar caracterele imposibil de afișat sunt raportate sub forma _.

  Argumentul $lungmax controlează lungimea maximă a rezultatului. Dacă rezultatul este mai lung decât $lungmax, va fi redus la $lungmax – 4 caractere și i se vor adăuga puncte de suspensie (...). Dacă $lungmax este 0, undef sau lipsește, valoarea sa prestabilită va fi $DBI::neat_maxlen(400).

  Nu folosiți metoda neat() pentru construcția interogărilor; dacă trebuie să executați încadrări între ghilimele sau modificări ale semnificației unor caractere speciale, folosiți în schimb metoda quote().

- $str = DBI::**neat_list**(\@listref [, $lungmax [, $sep_camp]]);

  Apelează metoda neat() pentru fiecare element din listă indicat de primul argument, îl unește cu șirul de separare $sep_camp și returnează rezultatul sub formă de șir unic.

  Argumentul $lungmax este aplicat argumentelor individuale, nu șirului care rezultă din apelurile la metoda neat().

  Dacă argumentul $sep_camp lipsește, valoarea sa prestabilită este ", ".

Atribute DBI

DBI furnizează informații despre atribute la numeroase nivele. Majoritatea atributelor sunt asociate cu variabile de manipulare a bazelor de date sau a instrucțiunilor, dar nu cu ambele. Unele atribute, precum PrintError și RaiseError, pot fi asociate cu variabile de manipulare a bazelor de date sau a instrucțiunilor. În general, fiecare variabilă de manipulare își are propriile sale atribute, dar unele atribute care conțin informații despre erori, precum err și errstr, au un caracter dinamic, în sensul că se asociază cu variabila de manipulare folosită cel mai recent.

Atribute generale ale variabilelor de manipulare

Aceste atribute por fi aplicate variabilelor de manipulare individuale sau pot fi specificate în parametrul %attr la metode care preiau un parametru pentru a controla modul de operare a metodei. Dacă aceste atribute sunt folosite cu metoda connect(), prelucrarea metodelor DBI este influențată global, pe toată întinderea scriptului.

- $h->{'**ChopBlanks**'}

  Determină dacă metodele de preluare a rândurilor vor amputa caracterele blanc finale ale valorilor din coloanele CHAR. Opțiunea ChopBlanks este dezactivată în mod prestabilit pentru majoritatea driverelor de baze de date, dar pentru MySQL starea acestei opțiuni nu are nici o importanță, deoarece serverul amputează întotdeauna caracterele blanc de la sfârșitul valorilor CHAR.

- $h->{'**PrintError**'}

  Dacă opțiunea este activată, apariția unei erori legate de DBI determină afișarea unui mesaj de avertizare. În mod prestabilit, opțiunea PrintError este activată.

- $h->{'**RaiseError**'}

  Dacă opțiunea este activată, apariția unei erori legate de DBI determină terminarea automată a execuției scriptului. În mod prestabilit, opțiunea PrintError este dezactivată.

Atribute dinamice

Aceste atribute sunt asociate variabilei de manipulare celei mai recent folosite, reprezentată prin expresia $h în descrierile următoare:

- **$DBI::err**

  Este o operație similară cu apelarea metodei $h->err().

- **$DBI::errstr**

  Este o operație similară cu apelarea metodei $h->errstr().

- **$DBI::rows**

  Este o operație similară cu apelarea metodei $h->rows().

Atribute ale variabilelor de manipulare a bazelor de date specifice sistemului MySQL

Aceste atribute sunt specifice driverului DBI pentru MySQL, DBD::mysql.

- $str = $dbh->{'**info**'};

  Acest atribut conține aceleași informații care sunt returnate de către funcția din interfața API C mysql_info(). Examinați descrierea acestei funcții în Anexa F.

- $rv = $dbh->{'**mysql_insertid**'};

  Valoarea AUTO_INCREMENT cel mai recent generată de conexiunea asociată cu $dbh.

- $rv = $dbh->{'**thread_id**'};

  Numărul de fir al conexiunii asociate cu $dbh.

Atribute ale variabilelor de manipulare a instrucțiunilor

În general, aceste atribute se aplică la o interogare SELECT (sau de tip SELECT) și nu sunt valide decât atunci când interogarea a fost transferată metodei prepare() pentru a obține o variabilă de manipulare a instrucțiunii și când a fost apelată metoda execute() pentru variabila respectivă. În plus, metoda finish() poate invalida unele atribute ale instrucțiunii.

Multe din aceste atribute au valori care constituie o referință la un tablou de valori, câte o valoare pentru fiecare coloană a interogării. Numărul de elemente al tabloului este dat de atributul $sth->{'NUM_OF_FIELDS'}. Pentru un atribut de instrucțiune atr_instr care este o referință la un tablou, puteți face referire la întregul tablou sub forma @{$sth->{atr_instr}} sau puteți parcurge ciclic elementele tabloului astfel:

```
for (my $i = 0; $i < $sth->{'NUM_OF_FIELDS'}; $i++)
{
   $value = $sth->{atr_instr}->[$i];
}
```

- **$sth->{'NAME'};**

  O referință la un tablou de șiruri care indică numele fiecărei coloane. Mărimea literelor folosite pentru nume (majuscule sau minuscule) este cea specificată în instrucțiunea SELECT.

- **$sth->{'NAME_lc'};**

  O referință la un tablou de șiruri care indică numele fiecărei coloane. Numele sunt returnate sub formă de șiruri scrise cu minuscule.

- **$sth->{'NAME_uc'};**

  O referință la un tablou de șiruri care indică numele fiecărei coloane. Numele sunt returnate sub formă de șiruri scrise cu majuscule.

- **$sth->{'NULLABLE'};**

  O referință la un tablou de șiruri care indică dacă fiecare coloană poate conține sau nu valoarea NULL. Valorile pentru fiecare element pot fi 0 (nu), 1 (da) sau 2 (necunoscut). Cel puțin așa se precizează în documentația DBI... Dacă în coloană nu pot exista valori NULL, se pare că valoarea returnată este de fapt un șir vid.

- **$sth->{'NUM_OF_FIELDS'};**

  Numărul de coloane care vor fi returnate de o instrucțiune pregătită, respectiv zero pentru o instrucțiune care nu este de tip SELECT.

- **$sth->{'NUM_OF_PARAMS'};**

  Numărul câmpurilor de înlocuire dintr-o instrucțiune pregătită.

- **$sth->{'PRECISION'};**

  O referință la un tablou de șiruri care indică precizia pentru fiecare coloană. DBI folosește termenul de „precizie" în sensul utilizat de ODBC, ceea ce pentru MySQL înseamnă lățimea maximă a coloanei. Pentru coloanele numerice, aceasta este lățimea de afișare. Pentru coloanele de tip șir, este lungimea maximă a coloanei.

- **$sth->{'SCALE'};**

  O referință la un tablou de șiruri care indică scara fiecărei coloane. DBI folosește termenul de „scară" în sensul utilizat de ODBC, ceea ce pentru MySQL înseamnă numărul cifrelor după virgulă, pentru coloanele cu virgulă mobilă. Pentru alte coloane, valoarea scării este 0.

- **$sth->{'STATEMENT'};**

  Textul instrucțiunii asociate cu $sth. Textul este cel văzut de metoda prepare() înainte de începerea operațiilor de substituire a câmpurilor de înlocuire.

- **$sth->{'TYPE'};**

  O referință la un tablou de valori care indică tipul numeric al fiecărei coloane.

Atribute ale variabilelor de manipulare a instrucțiunilor specifice sistemului MySQL

Majoritatea acestor atribute ale variabilelor de manipulare a instrucțiunilor trebuie să fie considerate ca fiind numai pentru citire, iar accesul la acestea trebuie să se realizeze după invocarea metodei execute(). Excepțiile sunt mysql_store_result și mysql_use_result. DBD::mysql furnizează posibilitatea de a controla stilul de prelucrare a setului de rezultate folosit de scriptul dumneavoastră. Atributele variabilelor de manipulare a instrucțiunilor mysql_store_result și mysql_use_result selectează maniera de prelucrare a setului de rezultate de către funcțiile din interfața API C mysql_store_result() și mysql_use_result(). Pentru o prezentare a acestor funcții și a diferențelor dintre ele, consultați Anexa F.

În mod prestabilit, DBI folosește mysql_store_result(), dar dumneavoastră puteți activa atributul mysql_use_result, care indică modulului DBI să folosească în schimb funcția mysql_use_result(). Următoarele operații trebuie executate după prepare(), dar înainte de execute():

```
$sth = $dbh->prepare (...);
$sth->{mysql_use_result} = 1;
$sth->execute();
```

Numeroase atribute specifice sistemului MySQL care erau disponibile în versiunile mai vechi ale driverului DBD::mysql sunt acum scoase din uz și au fost înlocuite de forme preferate mai recente, așa cum se poate vedea în tabelul G.2. Dacă versiunea dumneavoastră a driverului DBD::mysql este veche și nu acceptă atributele mai recente, încercați formele scoase din uz sau treceți la o versiune mai recentă.

Rețineți că insertid este un atribut al variabilelor de manipulare a instrucțiunilor, în timp ce forma sa mai recentă, mysql_insertid, este un atribut al variabilelor de manipulare a bazelor de date.

Tabelul G.2 Atribute specifice sistemului MySQL scoase din uz

Atribut scos din uz	Atribut preferat
insertid	mysql_insertid
is_blob	mysql_is_blob
is_key	mysql_is_key
is_not_null	mysql_is_not_null
is_num	mysql_is_num
is_pri_key	mysql_is_pri_key
length	PRECISION
max_length	mysql_max_length
table	mysql_table

- `$sth->{'mysql_is_blob'};`

  O referință la un tablou de valori care indică dacă fiecare coloană este sau nu de tip BLOB.

- `$sth->{'mysql_is_key'};`

  O referință la un tablou de valori care indică dacă fiecare coloană face sau nu parte dintr-o cheie non-unică.

- `$sth->{'mysql_is_not_null'};`

  O referință la un tablou de valori care indică dacă fiecare coloană poate conține sau nu valoarea NULL. O valoare „fals" arată că respectiva coloană poate conține valori NULL. Acest atribut conține informații care reprezintă „inversele" informațiilor conținute în atributul NULLABLE.

- `$sth->{'mysql_is_num'};`

  O referință la un tablou de valori care indică dacă fiecare coloană este sau nu de tip numeric.

- `$sth->{'mysql_is_pri_key'};`

  O referință la un tablou de valori care indică dacă fiecare coloană face sau nu parte dintr-o cheie primară (PRIMARY KEY).

- `$sth->{'mysql_max_length'};`

  O referință la un tablou de valori care indică lungimea maximă efectivă a valorilor coloanelor din setul de rezultate.

- `$sth->{'mysql_store_result'};`

  Dacă atributul mysql_store_result este activat (are valoarea 1), seturile de rezultate sunt regăsite din serverul MySQL care folosește funcția API C mysql_store_result(), nu funcția mysql_use_result(). Pentru o prezentare a acestor funcții și a diferențelor dintre ele, consultați Anexa F.

  Rețineți că utilizarea acestui atribut determină invalidarea unor atribute, între care și mysql_max_length. De asemenea, este invalidată utilizarea metodei rows(), deși este mai bine să numărați rândurile atunci când le preluați.

  În cazul în care activați atributul mysql_store_result, procedați astfel după ce ați invocat metoda prepare() și înainte de invocarea metodei execute().

- `$sth->{'mysql_table'}`

  O referință la un tablou de valori care indică numele tabelului din care provine fiecare coloană. În cazul unei coloane cu valori calculate, numele tabelului constă dintr-un șir vid.

- `$sth->{'mysql_type'}`

  O referință la un tablou de valori care indică numărul tipului MySQL pentru fiecare coloană din setul de rezultate.

- `$sth->{'mysql_type_name'}`

  O referință la un tablou de valori care indică numele tipului MySQL pentru fiecare coloană din setul de rezultate.

- `$sth->{'mysql_use_result'}`

  Dacă variabila mysql_use_result este activată (primește valoarea 1), seturile de rezultate sunt regăsite din serverul MySQL care folosește funcția API C mysql_use_result(), nu funcția mysql_store_result(). Pentru o prezentare a acestor funcții și a diferențelor dintre ele, consultați Anexa F.

  Rețineți că utilizarea acestui atribut determină invalidarea unor atribute precum mysql_max_length. De asemenea, este invalidată utilizarea metodelor rows(), deși se recomandă numărarea rândurilor în momentul preluării acestora.

  În cazul în care activați atributul mysql_use_result, procedați astfel după ce ați invocat metoda prepare() și înainte de invocarea metodei execute().

Variabile de mediu DBI

DBI consultă numeroase variabile de mediu, prezentate în tabelul G.3. Toate aceste variabile, cu excepția variabilei DBI_TRACE, sunt folosite de către metoda connect(). Variabila DBI_TRACE este folosită de către metoda trace().

Tabelul G.3 Variabile de mediu DBI

Nume	Semnificație
DBI_DRIVER	Numele driverului de nivel DBD ("mysql" pentru MySQL)
DBI_DSN	Numele sursei de date
DBI_PASS	Parolă
DBI_TRACE	Nivel de urmărire și/sau nivelul fișierului cu datele de ieșire ale procesului de urmărire
DBI_USER	Numele utilizatorului

ANEXA H

Referință API PHP

Această anexă descrie interfața PHP de programare a aplicațiilor pentru MySQL. Interfața API constă dintr-un set de funcții pentru comunicarea cu serverele MySQL și pentru accesul la bazele de date.

Această anexă servește drept referință, deci include numai scurte fragmente de program care ilustrează utilizarea funcțiilor din interfața API pentru PHP. Pentru scripturile client complete și note privind modul de redactare a acestora, vezi capitolul 8, „Interfața API pentru PHP". Funcțiile descrise aici sunt cele care au o legătură directă cu MySQL. Manualul PHP (care în prezent are aproape 800 de pagini) conține mai mult de 600 de pagini cu materiale de referință, deci este evident că această anexă trebuie să se limiteze la a rezuma o mică parte din caracteristicile PHP. Pentru a vă procura manualul PHP complet, vizitați situl Web PHP de la următoarea locație: `http://www.php.net/`

Redactarea scripturilor PHP

Scripturile PHP sunt fișiere text simplu, care pot conține o combinație între programe HTML și PHP. Scriptul este interpretat astfel încât să producă o pagină Web care este trimisă la client. Codul HTML este copiat la ieșire fără a fi interpretat. Codul PHP este interpretat și înlocuit cu datele de ieșire pe care le produce programul (care pot chiar lipsi).

PHP începe prin a interpreta un fișier în modul HTML. Puteți comuta în interiorul și în afara modului cod HTML folosind etichete speciale, care semnifică începutul și sfârșitul codului PHP. În interiorul unui fișier, puteți trece de la un mod la altul de oricâte ori doriți. PHP înțelege patru tipuri de etichete, deși unele dintre ele trebuie să fie activate în mod explicit, dacă doriți să le folosiți. O modalitate de activare a etichetelor constă în punerea lor în funcțiune în fișierul de inițializare PHP, `php3.ini`. Amplasamentul acestui fișier este dependent de sistem; deseori, acest fișier se găsește în `/usr/local/lib`.

PHP înțelege următoarele stiluri de etichete:

- Stilul prestabilit folosește etichetele `<?php` și `?>`:

```
<?php echo ("Ceva programe PHP pe-aici"); ?>
```

- Stilul cu etichetă de deschidere scurtă folosește etichetele `?` și `?`:

```
<? echo ("Ceva programe PHP pe-aici"); ?>
```

Acest stil de etichetă poate fi activat cu o directivă în fișierul de inițializare PHP:

```
short_open_tag = On;
```


- Stilul compatibil Active Server Page folosește etichetele `<%` și `%>`:

```
<% echo ("Ceva programe PHP pe-aici"); %>
```

Etichetele în stil ASP pot fi activate cu o directivă în fișierul de inițializare PHP:

```
asp_tags = On;
```

Suportul pentru etichetele ASP a fost introdus în versiunea PHP 3.0.4.

- Dacă folosiți un editor HTML care nu înțelege celelalte stiluri de etichete, puteți folosi etichetele `<SCRIPT>` și `</SCRIPT>`:

```
<SCRIPT LANGUAGE="php"> echo ("Ceva programe PHP pe-aici"); </SCRIPT>
```

Funcții

În continuare vom prezenta fiecare dintre funcțiile limbajului PHP legate de MySQL. Parametrii opționali sunt indicați prin paranteze drepte ([ ]).

Multe funcții preiau un parametru opțional `link_id` (identificator de legătură), care indică o conexiune cu un server MySQL. Dacă identificatorul de legătură lipsește dintr-un apel la o funcție, atunci se va folosi conexiunea cel mai recent deschisă. Dacă nu este specificată, respectiv nu este deschisă nici o conexiune, unele funcții vor încerca să stabilească o conexiune.

Un parametru `result_id` indică un identificator al setului de rezultate, returnat în mod caracteristic de funcțiile `mysql_db_query()` și `mysql_query()`.

Unele funcții generează un mesaj de eroare în cazul producerii unei erori, în afară de returnarea unei valori de stare. În contextele Web, acest mesaj apare în datele de ieșire trimise browserului client, ceea ce s-ar putea să nu intre în vederile dumneavoastră. Un nume de funcție poate fi precedat de caracterul `@` pentru a elimina mesajul de eroare (posibil criptic) pe care funcția l-ar genera în mod normal. De exemplu, pentru a elimina mesajul de eroare al unui apel la funcția `mysql_pconnect()` astfel încât să puteți raporta eșecul într-o manieră mai convenabilă, puteți proceda astfel:

```
<?php
$link = @mysql_pconnect ("pit-viper.snake.net","paul", "secret")
   or die ("Conectarea nu se poate realiza");
print ("Conectare reusita");
?>
```

O altă modalitate de a elimina mesajele de eroare este de a folosi funcția `error_reporting()`:

```
<?php
   error_reporting(0); # suprima toate mesajele de eroare
   $link = @mysql_pconnect ("pit-viper.snake.net", "paul", "secret")
      or die ("Conectarea nu se poate realiza");
   print ("Conectare reusita");
?>
```

Multe dintre scripturile date ca exemple în această anexă afișează mesajul `"Conectare reusita"` după stabilirea unei conexiuni cu serverul MySQL. Motivul pentru care procedează astfel este acela de a se asigura că scriptul afișează date de ieșire oarecare, în caz că doriți să încercați scriptul. Un script PHP care nu produce date de ieșire determină afișarea de către unele browsere a unui mesaj de avertizare de tipul „pagina nu conține date".

În descrierile prezentate în continuare, termenul de „interogare SELECT" trebuie asimilat atât cu o interogare SELECT, cât și cu orice altă interogare care returnează rânduri, precum DESCRIBE, EXPLAIN sau SHOW.

Rutine de gestiune a conexiunii

Rutinele din această secțiune vă permit să deschideți și să închideți conexiunile cu serverul MySQL.

- int

  mysql_close([int link_id]);

  Închide conexiunea cu serverul MySQL identificat de link_id. Dacă nu este specificată nici o conexiune, mysql_close() închide conexiunea cel mai recent deschisă.

  Funcția mysql_close() returnează „adevărat" în caz de reușită, respectiv „fals" la erori. Pentru conexiunile persistente deschise cu mysql_pconnect(), funcția mysql_close() ignoră cererea de închidere, dar returnează „adevărat". Dacă veți închide o conexiune, trebuie să o deschideți folosind mysql_connect() în locul funcției mysql_pconnect().

```
<?php
        $link = @mysql_pconnect ("pit-viper.snake.net", "paul", "secret")
                or die ("Conectarea nu se poate realiza");
        print ("Conectare reusita");
        mysql_close($link);
?>
```

- int

  mysql_connect([string nume_gazda] [, string nume_utilizator [, string parola]]);

  Deschide o conexiune cu serverul MySQL pe gazda nume_gazda pentru utilizatorul nume_utilizator având parola dată. Returnează identificatorul de legătură (un număr pozitiv) asociat cu noua conexiune; returnează „fals" în cazul producerii unei erori.

  Începând de la PHP 3.0.4, parametrul care desemnează numele gazdei poate fi specificat cu un număr de port opțional, în forma "nume_gazda:numar_port". Începând de la versiunea PHP 3.0.10, dacă numele gazdei este localhost, parametrul care indică numele gazdei poate fi specificat cu un nume de cale opțional pentru a preciza calea de acces a soclului de domeniu UNIX, sub forma "localhost:nume_soclu". Soclul trebuie specificat ca o cale de acces completă.

  Dacă parametrul care indică numele gazdei lipsește, valoarea prestabilită a acestuia este localhost. Dacă parametrul cu numele utilizatorului lipsește sau este vid, valoarea prestabilită este numele de utilizator sub contul căruia rulează PHP. (Acesta este numele utilizatorului sub contul căruia rulează serverul Web, dacă PHP rulează ca modul Apache, respectiv numele utilizatorului sub contul căruia rulează PHP, dacă PHP rulează ca program autonom.) Dacă parametrul parolă lipsește sau este vid, atunci se va trimite o parolă vidă.

  În timp ce conexiunea este deschisă, dacă funcția mysql_connect()este apelată cu aceiași parametri de conexiune (numele gazdei, numele de utilizator, parolă), nu se va genera o conexiune nouă; funcția mysql_connect() va returna în schimb identificatorul legăturii existente.

  Conexiunea poate fi închisă prin apelarea funcției mysql_close(). În cazul în care conexiunea este deschisă atunci când scriptul își încheie execuția, conexiunea va fi închisă automat.

```
<?php
        $link = @mysql_pconnect ("pit-viper.snake.net", "paul", "secret")
                or die ("Conectarea nu se poate realiza");
        print ("Conectare reusita");
        mysql_close($link);
?>
```

- int

  mysql_pconnect([string nume_gazda] [, string nume_utilizator [, string parola]]);

  Funcția mysql_pconnect() este asemănătoare funcției mysql_connect(), cu excepția faptului că deschide o conexiune persistentă. Cu alte cuvinte, conexiunea rămâne deschisă la terminarea execuției scriptului. Dacă se face un alt apel la funcția mysql_pconnect() cu aceiași parametri de conexiune (numele gazdei, numele de utilizator, parola) în timp ce conexiunea este deschisă, atunci conexiunea se va reutiliza. Astfel se evită suprasarcina de distrugere și redeschidere a conexiunilor, acest tip de conexiuni fiind mai eficient decât conexiunile non-persistente.

  Conexiunile persistente au sens numai atunci când PHP se execută ca modul în cadrul unui server Web care continuă să ruleze și după ce PHP și-a încheiat execuția. Într-un script executat de o versiune autonomă a sistemului PHP, conexiunea este închisă la terminarea scriptului, deoarece tot atunci se termină și procesul PHP.

  Apelarea funcției mysql_close() pe o conexiune persistentă este o operație lipsită de sens; în acest caz, funcția mysql_close() returnează „adevărat", dar lasă conexiunea deschisă.

```
<?php
        $link = @mysql_pconnect ("pit-viper.snake.net", "paul", "secret")
                or die ("Conectarea nu se poate realiza");
        print ("Conectare reusita");
?>
```

Rutine de raportare a stării și a erorilor

Funcțiile mysql_errno() și mysql_error() returnează numerele erorilor și mesaje de eroare pentru funcțiile PHP legate de MySQL. Cu toate acestea, nici una din funcții nu poate furniza informații privind erorile în absența unui identificator de legătură corect. Aceasta înseamnă că aceste funcții nu se pot utiliza pentru raportarea rezultatului unor apeluri ratate la funcțiile mysql_connect() sau mysql_pconnect(), deoarece nu există

nici un identificator de legătură decât atunci când s-a reușit stabilirea unei legături. Dacă doriți să obțineți mesajul de eroare MySQL pentru tentativele de conexiune eșuate, activați variabila `track_errors` cu o directivă în fișierul de inițializare PHP:

```
track_errors = On;
```

Apoi reporniți serverul Web, dacă rulați PHP ca modul Apache. După ce ați procedat astfel, puteți obține șirul de eroare pentru încercările de conexiune eșuate prin referirea la variabila `$php_errormsg`:

```
<?php
    $link = @mysql_connect ("gazda_rea", "utilizator_rau", "parola_rea")
            or die ("Conectare imposibila: " . $php_errormsg);
    print ("Conectare reusita");
?>
```

- `int`

  `mysql_errno``([int link_id]);`

  Pentru conexiunea dată, returnează numărul erorii pentru funcția conexă sistemului MySQL care a returnat cel mai recent un mesaj de eroare. O valoare egală cu zero indică faptul că nu s-a produs nici o eroare.

```
<?php
        $link = @mysql_pconnect ("pit-viper.snake.net", "paul", "secret")
            or die ("Conectarea nu se poate realiza");
        print ("Conectare reusita");
        $query = "SELECT * FROM presedinte";
        $result = mysql_query($query)
            or die ("interogare ratata, cod de eroare = " . mysql_errno());
?>
```

- `string`

  `mysql_error``([int link_id]);`

  Pentru conexiunea dată, returnează un șir care conține mesajul de eroare pentru funcția conexă sistemului MySQL care a returnat cel mai recent un mesaj de eroare. O valoare egală cu zero indică faptul că nu s-a produs nici o eroare.

```
<?php
        $link = @mysql_pconnect ("pit-viper.snake.net", "paul", "secret")
            or die ("Conectarea nu se poate realiza");
        print ("Conectare reusita");
        $query = "SELECT * FROM presedinte";
        $result = mysql_query($query)
            or die ("interogare ratata, mesaj de eroare = " .
                mysql_error());
?>
```


Rutine de construcție și execuție a interogărilor

Rutinele din această secțiune sunt folosite pentru a emite interogări spre serverul MySQL.

- `int`

  `mysql_db_query``(string nume_db, string interogare [, int link_id]);`

  Funcția `mysql_db_query()` este similară cu funcția `mysql_query()`, cu deosebirea că mai preia un argument suplimentar care indică numele unei baze de date, pe care o transformă în bază de date prestabilită înainte de a executa interogarea. (Comparați exemplul următor cu exemplul furnizat pentru funcția `mysql_query()`.)

```
<?php
        $link = @mysql_pconnect ("pit-viper.snake.net", "paul", "secret")
                or die ("Conectarea nu se poate realiza");
        print ("Conectare reusita");
        $query = "SELECT * FROM presedinte";
        $result = mysql_db_query("samp_db", $query)
                or die ("Interogare ratata");
?>
```

- `int`

  `mysql_list_dbs``([int link_id]);`

  Returnează un identificator de rezultat pentru un set de rezultate care constă din numele bazelor de date cunoscute de server pe conexiunea dată, câte un nume de bază de date în fiecare rând al setului de rezultate. Returnează „fals" dacă se produce o eroare. O bază de date prestabilită nu trebuie selectată. Setul de rezultate poate fi prelucrat de oricare dintre funcțiile obișnuite de preluare a rândurilor sau de către funcția `mysql_tablename()`.

```
<?php
        $link = @mysql_pconnect ("pit-viper.snake.net", "paul", "secret")
                or die ("Conectarea nu se poate realiza");
        $result = mysql_list_dbs()
                or die ("Interogare ratata");
        print ("Baze de date (folosind mysql_fetch_row()):<BR>\n";
        while ($row = mysql_fetch_row($result))
                printf("%s<BR>\n", $row[0]);
        $result = mysql_list_dbs()
                or die ("Interogare ratata");
        print ("Baze de date (folosind mysql_tablename()):<BR>\n";
        for ($i = 0; $i < mysql_num_rows($result); $i++)
                printf("%s<BR>\n", mysql_tablename($result, $i));
?>
```

- `int`

  `mysql_list_fields``(string nume_db, string nume_tabel [, int link_id]);`

Returnează un identificator de rezultat pentru un set de rezultate care conține informații despre coloanele dintr-un tabel; returnează „fals" dacă se produce o eroare. O bază de date prestabilită nu trebuie să fie selectată. Parametrii `nume_db` și `nume_tabel` identifică baza de date și tabelul care vă interesează. Identificatorul de rezultat se poate folosi cu oricare din funcțiile `mysql_field_flags()`, `mysql_field_len()`, `mysql_field_name()` și `mysql_field_type()`.

```
<?php
        $link = @mysql_pconnect ("pit-viper.snake.net", "paul", "secret")
                or die ("Conectarea nu se poate realiza");
        $result = mysql_list_fields("samp_db", "membru")
                or die ("Interogare ratata");
        print ("Informatii despre coloanele tabelului membru:BR\n";
        for ($i = 0; $i < mysql_num_fields($result);$i++)
        {
           printf("coloana %d:", $i);
           printf(" nume %s,\n", mysql_field_name($result, $i));
           printf(" lungime %d,\n", mysql_field_len($result, $i));
           printf(" tip %s,\n", mysql_field_type($result, $i));
           printf(" flags %s, \n", mysql_field_flags($result, $i));
           print("<BR>\n");
        }
?>
```

- int

mysql_list_tables(string nume_db [, int link_id]);

Returnează un identificator de rezultat pentru un set de rezultate care constă din numele tabelelor din baza de date specificată, câte un nume de tabel în fiecare rând al setului de rezultate. Returnează „fals" dacă se produce o eroare. O bază de date prestabilită nu trebuie selectată. Setul de rezultate poate fi prelucrat de oricare dintre funcțiile obișnuite de preluare a rândurilor sau de către funcția `mysql_tablename()`.

```
<?php
        $link = @mysql_pconnect ("pit-viper.snake.net", "paul", "secret")
                or die ("Conectarea nu se poate realiza");
        $result = mysql_list_tables("samp_db")
                or die ("Interogare ratata");
        print ("tabelele din samp_db (cu mysql_fetch_row()):<BR>\n";
        while ($row = mysql_fetch_row($result))
                printf("%s<BR>\n", $row[0]);
        $result = mysql_list_tables("samp_db")
                or die ("Interogare ratata");
        print ("tabelele din samp_db (cu mysql_tablename()):<BR>\n";
        for ($i = 0; $i < mysql_num_rows($result); $i++)
                printf("%s<BR>\n", mysql_tablename($result, $i));
?>
```


- int

mysql_query(string interogare [, int link_id]);

Trimite șirul de interogare către serverul MySQL pe conexiunea dată. Pentru instrucțiunile `DELETE`, `INSERT`, `REPLACE` și `UPDATE`, funcția `mysql_query()` returnează „adevărat" pentru succes, respectiv „fals" dacă se produce o eroare. Pentru o interogare reușită, puteți apela funcția `mysql_affected_rows()` pentru a descoperi numărul rândurilor modificate.

Pentru instrucțiunile `SELECT`, funcția `mysql_query()` returnează un identificator sigur al setului de rezultate în caz de reușită, respectiv „fals" dacă se produce vreo eroare. Pentru o interogare reușită, identificatorul de rezultat poate fi folosit cu diferitele funcții de prelucrare a setului de rezultate care preiau un argument `result_id`. Identificatorul poate fi transferat funcției `mysql_free_result()` pentru eliberarea tuturor resurselor asociate setului de rezultate.

O interogare „reușită" se execută fără erori, dar reușita nu implică nimic legat de returnarea sau nereturnarea unor rânduri. Interogarea următoare este absolut corectă, dar nu returnează nici un rând:

```
SELECT * FROM presedinte WHERE 1 = 0
```

O interogare poate eșua din numeroase motive. De exemplu, poate fi eronată sintactic, incorectă din punct de vedere semantic sau ilegală, deoarece nu aveți permisiunea de a obține acces la tabelele specificate în interogare.

Dacă nu este specificat nici un identificator de legătură, se va folosi conexiunea cel mai recent deschisă. Dacă nu există nici o conexiune curentă, funcția `mysql_query()` încearcă să deschidă o conexiune ca și cum funcția `mysql_connect()` ar fi fost apelată fără argumente. Funcția `mysql_query()` eșuează dacă încercarea de conectare nu reușește.

```
<?php
        $link = @mysql_pconnect ("pit-viper.snake.net", "paul", "secret")
                or die ("Conectarea nu se poate realiza");
        print ("Conectare reusita");
        mysql_select_db("samp_db")
                or die ("Nu poate selecta baza de date");
        $query = "SELECT * FROM presedinte";
        $result = mysql_query($query)
                or die ("Interogare ratata");
?>
```

Rutine de prelucrare a setului de rezultate

Rutinele din această secțiune sunt folosite pentru regăsirea rezultatelor interogărilor. De asemenea, furnizează acces la informații despre rezultat, precum numărul rândurilor afectate sau metadatele aferente coloanelor setului de rezultate.

- int

mysql_affected_rows([int link_id]);

Returnează numărul rândurilor afectate (modificate) de către cea mai recentă instrucțiune DELETE, INSERT, REPLACE sau UPDATE executată pe conexiunea dată. Funcția mysql_affected_rows() returnează 0 dacă nu au fost modificate rânduri, respectiv –1 dacă s-a produs o eroare.

După o interogare SELECT, funcția mysql_affected_rows() returnează numărul rândurilor selectate. Totuși, la instrucțiunile SELECT se folosește de obicei funcția mysql_num_rows().

```
<?php
        $link = @mysql_pconnect ("pit-viper.snake.net", "paul", "secret")
                or die ("Conectarea nu se poate realiza");
        mysql_select_db("samp_db")
                or die ("Nu poate selecta baza de date");
        $query = "INSERT INTO membru (nume,prenume,data_expirare)"
                . " VALUES('Vasile','Cornel','2002-6-3');
        $result = mysql_query($query)
                or die ("Interogare ratata");
        printf ("%d randuri inserate\n",
                        mysql_affected_rows(),
                        mysql_affected_rows() = = 1 ? "" : "s");
?>
```

- int

mysql_data_seek(int result_id, int row_num);

Fiecare set de rezultate returnate de o interogare SELECT are un cursor de rând, care indică rândul ce trebuie returnat de următorul apel la funcțiile de preluare a rândurilor, în speță mysql_fetch_array(), mysql_fetch_object() sau mysql_fetch_row(). Funcția mysql_data_seek() fixează cursorul pentru setul de rezultate dat pe rândul dat. Numărul rândului trebuie să fie cuprins între 0 și mysql_num_rows()–1. Funcția mysql_data_seek() returnează „adevărat" dacă numărul rândului este corect, respectiv „fals" în caz contrar.

```
<?php
        $link = @mysql_pconnect ("pit-viper.snake.net", "paul", "secret")
                or die ("Conectarea nu se poate realiza");
        mysql_select_db("samp_db")
                or die ("Nu poate selecta baza de date");
        $query = "SELECT nume,prenume FROM presedinte";
        $result = mysql_query($query)
                or die ("Interogare ratata");
        # preia randurile in ordine inversa
        for ($i = mysql_num_rows($result) – 1; $i >= 0; $i-)
        {
                if (!mysql_data_seek($result, $i))
                {
                        printf("Nu poate cauta in randul %d\n", $i);
                        continue;
```



```
                }
                if(!($row = mysql_fetch_object($result)))
                        continue;
                printf("%s %s<BR>\n", $row->nume, $row->prenume);
        }
        mysql_free_result($result);
?>
```

- array

mysql_fetch_array(int result_id, [, int result_type]);

Returnează următorul rând al setului de rezultate date sub forma unui tablou. Returnează „fals" dacă nu mai există rânduri. Tabloul conține valori stocate atât sub formă de indici numerici de coloană, cât și sub formă de chei asociative determinate de numele coloanelor. Cu alte cuvinte, fiecare valoare dintr-o coloană este accesibilă atât prin indexul său numeric, cât și prin numele său. Indicii asociativi sunt sensibili la diferența între majuscule și minuscule și trebuie scriși cu aceeași mărime de literă folosită pentru desemnarea coloanelor în cadrul interogării. Să presupunem că emiteți următoarea interogare:

```
SELECT nume,prenume FROM presedinte
```

Dacă preluați rândurile dintr-un set de rezultate într-un tablou denumit $row, elementele tabloului sunt accesibile după cum urmează:

$row[0]	Contine valoarea nume
$row[1]	Contine valoarea prenume
$row["nume"]	Contine valoarea nume
$row["prenume"]	Contine valoarea prenume

Cheile nu conțin numele tabelelor pentru coloanele corespunzătoare, deci, dacă selectați coloane cu același nume din tabele diferite, veți obține un conflict de nume. Prioritatea este acordată coloanei specificate ultima în lista de coloane specificată de interogare. Pentru a obține accesul la coloana ascunsă, folosiți indexul numeric al acesteia sau scrieți interogarea pentru a furniza coloanei un alias.

Parametrul result_type poate fi MYSQL_ASSOC (returnează valorile numai în funcție de indicii de nume), MYSQL_NUM (returnează valorile numai în funcție de indicii numerici), respectiv MYSQL_BOTH (returnează valorile în funcție de ambele categorii de indici). Dacă result_type nu este specificat, valoarea prestabilită este MYSQL_BOTH.

```
<?php
        $link = @mysql_pconnect ("pit-viper.snake.net", "paul", "secret")
                or die ("Conectarea nu se poate realiza");
        mysql_select_db("samp_db")
                or die ("Nu poate selecta baza de date");
        $query = "SELECT nume,prenume FROM presedinte";
        $result = mysql_query($query)
                or die ("Interogare ratata");
while ($row = mysql_fetch_array($result))
```

Continuare

Continuare

```
{
        # afiseaza fiecare nume de doua ori, o data folosind
        # indici numerici, iar alta data folosind indici
        # asociativi (de nume)
        printf("%s %s<BR>\n", $row[0], $row[1]);
        printf("%s %s<BR>\n", $row["nume"], $row["prenume"]);
}
mysql_free_result();
?>
```

- object

mysql_fetch_field(int result_id [, int col_num]);

Returnează metadatele referitoare la coloana dată din setul de rezultate; returnează „fals" dacă nu există o asemenea coloană. Dacă parametrul col_num este omis, apelurile succesive la funcția mysql_fetch_field() returnează informații despre coloanele succesive ale setului de rezultate. Valoarea returnată este „fals" dacă nu au mai rămas coloane. Dacă parametrul col_num este specificat, valoarea acestuia trebuie să fie cuprinsă între 0 și mysql_num_fields()-1. În acest caz, mysql_fetch_field() returnează informații despre coloana dată, respectiv „fals" dacă valoarea parametrului col_num nu se află în domeniul specificat.

Informațiile sunt returnate sub forma unui obiect care are proprietățile prezentate în tabelul H.1.

```
<?php
        $link = @mysql_pconnect ("pit-viper.snake.net", "paul", "secret")
                or die ("Conectarea nu se poate realiza");
        mysql_select_db("samp_db")
                or die ("Nu poate selecta baza de date");
        $query = "SELECT * FROM presedinte";
        $result = mysql_query($query)
                or die ("Interogare ratata");
        # preia metadatele coloanei
        for ($i = 0; $i < mysql_num_fields($result); $i++)
        {
                printf("Informatii pentru coloana %d:<BR>\n", $i);
                $meta = mysql_fetch_field($result);
                if(!$meta)
                {
                        print("Nu exista informatii<BR>\n");
                        continue;
                }
                print("<PRE>\n");
                printf("blob:           %s/n", $meta->blob);
                printf("max_length:     %s/n", $meta->max_length);
```



```
                printf("multiple_key:   %s/n", $meta->multiple_key);
                printf("name:           %s/n", $meta->name);
                printf("not_null:       %s/n", $meta->not_null);
                printf("numeric:        %s/n", $meta->numeric);
                printf("primary_key:    %s/n", $meta->primary_key);
                printf("table:          %s/n", $meta->table);
                printf("type:           %s/n", $meta->type);
                printf("unique_key:     %s/n", $meta->unique_key);
                printf("unsigned:       %s/n", $meta->unsigned);
                printf("zerofill:       %s/n", $meta->zerofill);
                print("</PRE>\n");
        }
?>
```

Tabelul H.1 Proprietățile obiectului mysql_fetch_field()

Proprietate	Semnificație
blob	1, în cazul în care coloana este de tip BLOB, 0 în caz contrar
max_length	Lungimea celei mai mari valori din coloanele setului de rezultate
multiple_key	1, în cazul în care coloana face parte dintr-un index non-unic, 0 în caz contrar
name	Numele coloanei
not_null	1 în cazul în care coloana nu poate conține valori NULL, 0 în caz contrar
numeric	1 în cazul în care coloana este de tip numeric, 0 în caz contrar
primary_key	1 în cazul în care coloana face parte dintr-o cheie primară, 0 în caz contrar
table	Numele tabelului care conține coloana (are valoare vidă, în cazul în care coloana conține valori calculate)
type	Numele tipului coloanei
unique_key	1 în cazul în care coloana face parte dintr-un index UNIQUE, 0 în caz contrar
unsigned	1 în cazul în care coloana are atributul UNSIGNED, 0 în caz contrar
zerofill	1 în cazul în care coloana are atributul ZEROFILL, 0 în caz contrar

- array

mysql_fetch_lengths(int result_id);

Returnează un tablou care conține lungimile valorilor coloanelor din rândul cel mai recent preluat de oricare dintre funcțiile mysql_fetch_array(), mysql_fetch_object() sau mysql_fetch_row(). Returnează „fals" dacă nu a fost încă preluat nici un rând, respectiv dacă s-a produs o eroare.

```
<?php
        $link = @mysql_pconnect ("pit-viper.snake.net", "paul", "secret")
                or die ("Conectarea nu se poate realiza");
        mysql_select_db("samp_db")
```

Continuare

Continuare

```
                or die ("Nu poate selecta baza de date");
        $query = "SELECT * FROM presedinte";
        $result = mysql_query($query)
                or die ("Interogare ratata");
$row_num = 0;
while (mysql_fetch_row(result))
{
        ++$row_num;
        # obtine lungimile valorilor din coloane
        printf("Lungimile valorilor din randul %d:<BR>\n", $row_num);
        $len = mysql_fetch_lengths($result);
        if(!$len)
        {
                print("Nu exista informatii<BR>\n");
                break;
        }
        print("<PRE>\n");
        for ($i = 0; $i < mysql_num_fields($result); $i++)
                printf("coloana %d: %s\n", $i, $len[$i]);
        print("</PRE>\n");
}
?>
```

- object

mysql_fetch_object(int result_id, [, int result_type]);

Returnează rândul următor al setului de rezultate dat sub formă de obiect; returnează „fals" dacă nu au mai rămas rânduri. Valorile coloanelor sunt accesibile sub formă de proprietăți ale obiectului. Numele proprietăților sunt numele coloanelor selectate în interogarea care a generat setul de rezultate.

Parametrul result_type poate fi MYSQL_ASSOC (returnează valorile numai în funcție de indicii de nume), MYSQL_NUM (returnează valorile numai în funcție de indicii numerici), respectiv MYSQL_BOTH (returnează valorile în funcție de ambele categorii de indici). Dacă result_type nu este specificat, valoarea prestabilită este MYSQL_BOTH. (Mie unuia nu-mi este clar care este rostul specificării opțiunii MYSQL_NUM, din moment ce numerele nu pot constitui nume acceptate de proprietăți.)

```
<?php
        $link = @mysql_pconnect ("pit-viper.snake.net", "paul", "secret")
                or die ("Conectarea nu se poate realiza");
        mysql_select_db("samp_db")
                or die ("Nu poate selecta baza de date");
        $query = "SELECT nume,prenum FROM presedinte";
        $result = mysql_query($query)
```



```
                or die ("Interogare ratata");
        while ($row = mysql_fetch_object($result))
                printf("%s %s<BR>\n", $row->nume, $row->prenume);
        mysql_free_result($result);
?>
```

- array

mysql_fetch_row(int result_id);

Returnează rândul următor al setului de rezultate sub formă de tablou; returnează „fals" dacă nu mai există rânduri.

Valorile din coloane sunt accesibile ca elemente ale tabloului, folosind indici de coloană cu valori cuprinse între 0 și mysql_num_fields()-1.

```
<?php
        $link = @mysql_pconnect ("pit-viper.snake.net", "paul", "secret")
                or die ("Conectarea nu se poate realiza");
        mysql_select_db("samp_db")
                or die ("Nu poate selecta baza de date");
        $query = "SELECT nume,prenume FROM presedinte";
        $result = mysql_query($query)
                or die ("Interogare ratata");
        while ($row = mysql_fetch_row($result))
                printf("%s %s<BR>\n", $row[0], $row[1]);
        mysql_free_result($result);
?>
```

- string

mysql_field_name(int result_id, int col_num);

Returnează numele coloanei date din setul de rezultate. Valoarea parametrului col_num trebuie să fie cuprinsă între 0 și mysql_num_fields()-1.

```
<?php
        $link = @mysql_pconnect ("pit-viper.snake.net", "paul", "secret")
                or die ("Conectarea nu se poate realiza");
        mysql_select_db("samp_db")
                or die ("Nu poate selecta baza de date");
        $query = "SELECT * FROM presedinte";
        $result = mysql_query($query)
                or die ("Interogare ratata");
        # obtine numele coloanelor
        for ($i = 0; $i < mysql_num_fields($result); $i++)
        {
                printf("Numele coloanei %d: ", $i);
                $name = mysql_field_name($result, $i);
                if(!$name)
```

Continuare

Continuare

```
                print("Nu exista nici un nume<BR>\n");
        else
                print("$name<BR>\n");
    }
?>
```

- int

mysql_field_seek(int result_id, int col_num);

Stabilește indexul pentru apelurile ulterioare la funcția mysql_fetch_field(). Următorul apel către funcția mysql_fetch_field() care este emis fără un număr de coloană explicit va returna informații pentru coloana col_num. Returnează „adevărat" în cazul în care căutarea reușește și „fals" în caz contrar. Valoarea parametrului col_num trebuie să fie cuprinsă între 0 și mysql_num_fields()-1.

```
<?php
        $link = @mysql_pconnect ("pit-viper.snake.net", "paul", "secret")
                or die ("Conectarea nu se poate realiza");
        mysql_select_db("samp_db")
                or die ("Nu poate selecta baza de date");
        $query = "SELECT * FROM presedinte";
        $result = mysql_query($query)
                or die ("Interogare ratata");
        # preia metadatele coloanei
        for ($i = 0; $i < mysql_num_fields($result); $i++)
        {
                printf("Informatiile pentru coloana %d:<BR>\n", $i);
                if (!mysql_field_seek($result, $i))
                {
                        print("Nu poate cauta in coloana<BR>\n");
                        continue;
                }
                $meta = mysql_fetch_field($result, $i);
                if(!$meta)
                {
                        print("Nu exista informatii<BR>\n");
                        continue;
                }
                print("<PRE>\n");
                printf("blob:           %s/n", $meta->blob);
                printf("max_length:     %s/n", $meta->max_length);
                printf("multiple_key:   %s/n", $meta->multiple_key);
                printf("name:           %s/n", $meta->name);
                printf("not_null:       %s/n", $meta->not_null);
                printf("numeric:        %s/n", $meta->numeric);
```



```
                printf("primary_key:    %s/n", $meta->primary_key);
                printf("table:          %s/n", $meta->table);
                printf("type:           %s/n", $meta->type);
                printf("unique_key:     %s/n", $meta->unique_key);
                printf("unsigned:       %s/n", $meta->unsigned);
                printf("zerofill:       %s/n", $meta->zerofill);
                print("</PRE>\n");
        }
?>
```

- string

mysql_field_table(int result_id, int col_num);

Returnează numele tabelului care conține coloana dată a setului de rezultate. Pentru coloanele care conțin valori calculate, numele este vid.

Valoarea parametrului col_num trebuie să fie cuprinsă între 0 și mysql_num_fields()-1.

```
<?php
        $link = @mysql_pconnect ("pit-viper.snake.net", "paul", "secret")
                or die ("Conectarea nu se poate realiza");
        mysql_select_db("samp_db")
                or die ("Nu poate selecta baza de date");
        $query = "SELECT * FROM presedinte";
        $result = mysql_query($query)
                or die ("Interogare ratata");
        for ($i = 0; $i < mysql_num_fields($result); $i++)
        {
                printf("coloana %d:", $i);
                printf(" nume %s,\n", mysql_field_name($result, $i));
                printf(" tabel %s,\n", mysql_field_table($result, $i));
                print "<BR>\n";
        }
?>
```

- string

mysql_field_type(int result_id, int col_num);

Returnează numele tipului coloanei date a setului de rezultate. Numele tipurilor de date sunt enumerate în anexa B, „Referință de tipuri de coloane".

Valoarea parametrului col_num trebuie să fie cuprinsă între 0 și mysql_num_fields()-1.

```
<?php
        $link = @mysql_pconnect ("pit-viper.snake.net", "paul", "secret")
                or die ("Conectarea nu se poate realiza");
        mysql_select_db("samp_db")
                or die ("Nu poate selecta baza de date");
        $query = "SELECT * FROM presedinte";
```

Continuare

Continuare

```
        $result = mysql_query($query)
                or die ("Interogare ratata");
        for ($i = 0; $i < mysql_num_fields($result); $i++)
        {
                printf("coloana %d:", $i);
                printf(" nume %s,\n", mysql_field_name($result, $i));
                printf(" tip %s,\n", mysql_field_type($result, $i));
                print "<BR>\n";
        }
?>
```

- string

  mysql_field_flags(int result_id, int col_num);

  Returnează metadatele aferente coloanei date din setul de rezultate, sub formă de șir; returnează „fals" dacă se produce o eroare. Șirul constă din cuvinte separate prin spații, care indică dacă valorile indicator ale coloanei sunt „adevărat". Pentru indicatoarele cu valoarea „fals", cuvântul corespunzător nu se găsește în șir. Tabelul H.2 enumeră cuvintele care se pot găsi în șir.

  Valoarea parametrului col_num trebuie să fie cuprinsă între 0 și mysql_num_fields()-1.

```
<?php
        $link = @mysql_pconnect ("pit-viper.snake.net", "paul", "secret")
                or die ("Conectarea nu se poate realiza");
        mysql_select_db("samp_db");
        $query = "SELECT * FROM membru";
        $result = mysql_query($query)
                or die ("Interogare ratata");
        for ($i = 0; $i < mysql_num_fields($result); $i++)
        {
                printf("coloana %d:", $i);
                printf(" nume %s,\n", mysql_field_name($result, $i));
                printf(" flags %s,\n", mysql_field_flags($result, $i));
                print "<BR>\n";
        }
?>
```

Tabelul H.2 Valorile returnate de funcția mysql_field_flags()

Proprietate	Semnificație
auto_increment	Coloana are atributul AUTO_INCREMENT
binary	Coloana are atributul BINARY
blob	Coloana este de tip BLOB
enum	Coloana este de tip ENUM

Proprietate	Semnificație
multiple_key	Coloana face parte dintr-un index non-unic
not_null	Coloana nu poate conține valori NULL
primary_key	Coloana face parte dintr-o cheie primară
timestamp	Coloana este de tip TIMESTAMP
unique_key	Coloana face parte dintr-un index UNIQUE
unsigned	Coloana are atributul UNSIGNED
zerofill	Coloana are atributul ZEROFILL

- int

  mysql_field_len(int result_id, int col_num);

  Returnează lungimea maximă posibilă a valorilor din coloana dată a setului de rezultate. Valoarea parametrului col_num trebuie să fie cuprinsă între 0 și mysql_num_fields()-1.

```
<?php
        $link = @mysql_pconnect ("pit-viper.snake.net", "paul", "secret")
                or die ("Conectarea nu se poate realiza");
        mysql_select_db("samp_db");
        $query = "SELECT * FROM membru";
        $result = mysql_query($query)
                or die ("Interogare ratata");
        for ($i = 0; $i < mysql_num_fields($result); $i++)
        {
                printf("coloana %d:", $i);
                printf(" nume %s,\n", mysql_field_name($result, $i));
                printf(" lungime %d,\n", mysql_field_len($result, $i));
                print "<BR>\n";
        }
?>
```

- int

  mysql_free_result(int result_id);

  Eliberează toate resursele asociate setului de rezultate dat. Seturile de rezultate sunt eliberate automat la terminarea execuției unui script, dar puteți dori să apelați această funcție în mod explicit, într-un script care generează numeroase seturi de rezultate. De exemplu, acest script va folosi o cantitate considerabilă de memorie:

```
<?php
        $link = mysql_connect("localhost","paul","secret");
        mysql_select_db("samp_db");
        for ($i = 0; $i < 10000; $i++)
        {
                $result = mysql_query("SELECT * FROM presedinte");
        }
?>
```

Dacă se adaugă un apel la funcția mysql_free_result() după apelul la funcția mysql_query(), se va reduce cantitatea de memorie utilizată pentru stocarea setului de rezultate până aproape de zero:

```
<?php
        $link = mysql_connect("localhost","paul","secret");
        mysql_select_db("samp_db");
        for ($i = 0; $i < 10000; $i++)
        {
                $result = mysql_query("SELECT * FROM presedinte");
                mysql_free_result($result);
        }
?>
```

- int

mysql_insert_id([int link_id]);

Returnează valoarea AUTO_INCREMENT generată de interogarea cel mai recent executată pe conexiunea dată. Returnează zero dacă pe durata vieții conexiunii respective nu a fost generată nici o asemenea valoare. În general, trebuie să apelați funcția mysql_insert_id() imediat după o interogare care se presupune că va genera o valoare nouă; dacă orice altă interogare se interpune între interogarea respectivă și locul unde doriți să folosiți valoarea în cauză, valoarea funcției mysql_insert_id() poate fi reinițializată la zero de către interogarea intercalată.

Rețineți că modul de comportare a funcției mysql_insert_id() diferă de acela al funcției SQL LAST_INSERT_ID(). Funcția mysql_insert_id() este menținută la client și valoarea sa este stabilită pentru fiecare interogare în parte. Valoarea funcției LAST_INSERT_ID()este menținută în server și se păstrează de la o interogare la alta.

Valorile returnate de funcția mysql_insert_id() sunt specifice conexiunii și nu sunt afectate de activitatea AUTO_INCREMENT desfășurată pe alte conexiuni.

```
<?php
        $link = @mysql_pconnect ("pit-viper.snake.net", "paul", "secret")
                or die ("Conectarea nu se poate realiza");
        mysql_select_db("samp_db")
                or die ("Nu poate selecta baza de date");
        $query = "INSERT INTO membru (nume,prenume,data_expirare)"
                . " VALUES('Vasile','Cornel','2002-6-3')";
        $result = mysql_query($query)
                or die ("Interogare ratata");
        printf("numarul noului membru: %d\n", mysql_insert_id());
?>
```

- int

mysql_num_fields(int result_id);

Returnează numărul de coloane din setul de rezultate dat.


```
<?php
        $link = @mysql_pconnect ("pit-viper.snake.net", "paul", "secret")
                or die ("Nu poate selecta baza de date");
        mysql_select_db("samp_db");
        $query = "SELECT * FROM presedinte";
        $result = mysql_query($query)
                or die ("Interogare ratata");
        printf("Numarul de coloane: %d\n", mysql_num_fields($result));
?>
```

- int

mysql_num_rows(int result_id);

Returnează numărul de rânduri din setul de rezultate dat.

```
<?php
        $link = @mysql_pconnect ("pit-viper.snake.net", "paul", "secret")
                or die ("Nu poate selecta baza de date");
        mysql_select_db("samp_db");
        $query = "SELECT * FROM presedinte";
        $result = mysql_query($query)
                or die ("Interogare ratata");
        printf("Numarul de randuri: %d\n", mysql_num_rows($result));
?>
```

- int

mysql_result(int result_id, int row, mixed field);

Returnează o valoare din rândul dat al unui set de rezultate. Coloana este identificată prin parametrul field, care poate fi un index numeric de coloană sau numele coloanei specificat în interogare.

Această funcție este lentă; se recomandă utilizarea uneia din funcțiile mysql_fetch_array(), mysql_fetch_object() sau mysql_fetch_row().

```
<?php
        $link = @mysql_pconnect ("pit-viper.snake.net", "paul", "secret")
                or die ("Conectarea nu se poate realiza");
        mysql_select_db("samp_db")
                or die ("Nu poate selecta baza de date");
        $query = "SELECT nume,prenume FROM presedinte";
        $result = mysql_query($query)
                or die ("Interogare ratata);
        for ($i = 0; $i < mysql_num_rows($result); $i++)
        {
                for ($j = 0; $j < mysql_num_rows($result); $j++)
                {
                        if($j < 0)
                                print(" ");
```

Continuare

Continuare

```
                print(mysql_result($result, $i, $j));
            }
            print "<BR>\n";
        }
        mysql_free_result($result);
?>
```

- string

mysql_tablename(int result_id, int row_num);

Dacă este dat un identificator de rezultat returnat de mysql_list_dbs()sau mysql_list_tables(), precum și un index de rând row_num, funcția returnează numele stocat în rândul dat al setului de rezultate.

Valoarea indexului de rând trebuie să fie cuprinsă între 0 și mysql_num_rows()-1.

```
<?php
        $link = @mysql_pconnect ("pit-viper.snake.net", "paul", "secret")
                or die ("Conectarea nu se poate realiza");
        $result = mysql_list_tables("samp_db")
                or die ("Interogare ratata");
        print("tabelele din baza de date samp_db:<BR>\n");
        for ($i = 0; $i < mysql_num_rows($result); $i++)
                printf("%s<BR>\n", mysql_tablename($result, $i));
?>
```

Rutine pentru baze de date

Rutinele din această secțiune vă permit să creați și să distrugeți bazele de date. De asemenea, vă permit să selectați o bază de date prestabilită pentru sesiunea server curentă.

- int

mysql_create_db(string nume_db, [, int link_id]);

Indică serverului MySQL identificat prin link_id să creeze baza de date cu numele dat. Returnează „adevărat" dacă baza de date a fost creată cu succes, respectiv „fals" dacă s-a produs o eroare. Pentru a crea baza de date, trebuie să dispuneți de privilegiul CREATE aferent acesteia.

Este preferabil să folosiți funcția mysql_query() pentru a emite o instrucțiune CREATE DATABASE, decât să folosiți funcția mysql_create_db().

```
<?php
        $link = @mysql_pconnect ("pit-viper.snake.net", "paul", "secret")
                or die ("Conectarea nu se poate realiza");
        if (mysql_create_db("my_db"))
                print("Baza de date a fost creata cu succes\n");
        else
                printf("Eroare la crearea bazei de date: %s\n",
                        mysql_error());
?>
```


- int

mysql_drop_db(string nume_db, [, int link_id]);

Indică serverului MySQL identificat prin link_id să șteargă baza de date cu numele dat. Returnează „adevărat" dacă baza de date a fost eliminată cu succes, respectiv „fals" dacă s-a produs o eroare. Pentru a elimina baza de date, trebuie să dispuneți de privilegiul DROP aferent acesteia.

Fiți atent cu această funcție; dacă ștergeți o bază de date, ștearsă rămâne. Nu o mai puteți recupera.

Este preferabil să folosiți funcția mysql_query() pentru a emite o instrucțiune DROP DATABASE, decât să folosiți funcția mysql_drop_db().

```
<?php
        $link = @mysql_pconnect ("pit-viper.snake.net", "paul", "secret")
                or die ("Conectarea nu se poate realiza");
        if (mysql_drop_db("my_db"))
                print("Baza de date a fost eliminata cu succes\n");
        else
                printf("Eroare la eliminarea bazei de date: %s\n",
                        mysql_error());
?>
```

- int

mysql_select_db(string nume_db, [, int link_id]);

Selectează baza de date specificată pentru a o transforma în bază de date prestabilită pentru conexiunea dată. Returnează „adevărat" în caz de succes, respectiv „fals" dacă se produce o eroare.

Dacă nu este specificată nici o conexiune, se va folosi conexiunea curentă. Dacă nu există o conexiune curentă, funcția mysql_select_db() încearcă să deschidă o conexiune ca și cum funcția mysql_connect() ar fi fost apelată fără argumente. Funcția mysql_select_db() eșuează dacă încercarea de conexiune nu reușește.

```
<?php
        $link = @mysql_pconnect ("pit-viper.snake.net", "paul", "secret")
                or die ("Conectarea nu se poate realiza");
        print ("Conectarea s-a realizat cu succes");
        mysql_select_db("samp_db")
                or die ("Nu s-a putut selecta baza de date");
?>
```

ANEXA I

Instrumente utile produse de terțe părți

Această anexă vă oferă o succintă vedere de ansamblu ilustrativă a tipurilor de instrumente create de terțe părți disponibile care vă pot ajuta să folosiți sistemul MySQL mai simplu și mai eficient. O listă cu programe mai cuprinzătoare și frecvent actualizată este disponibilă în situl Web al sistemului MySQL. Intenția anexei de față este pur și simplu de a vă pune la curent cu unele dintre tipurile de instrumente disponibile și de a vă încuraja să examinați personal câteva dintre acestea. Pentru mai multe informații, vizitați adresa `http://www.mysql.com/Contrib/`. Veți descoperi numeroase pachete care vă pot fi de ajutor. Aceste pachete sunt grupate în mai multe categorii:

- **Interfețe de programare a aplicațiilor în diverse limbaje și instrumente de programare.** MySQL este bine reprezentat de către interfețele de programare. În afară de interfețele Perl și PHP care au fost discutate în capitolele 7, „Interfața API pentru Perl DBI" și 8, „Interfața API pentru PHP", sunt disponibile interfețe pentru numeroase alte limbaje. Acestea includ C++, Delphi, Eiffel, Guile, Java, Matlab, Pike, Python, Ruby, Smalltalk și Tcl. Există și biblioteci container, cum este o bibliotecă C care prezintă o interfață în stil Visual Basic cu programul gazdă.

  Există și alte instrumente care vă ajută să folosiți interfețe API. Puteți prelua date de ieșire produse de `mysqldump` care descriu structuri de date și le puteți converti într-un fișier antet C. Acest fapt este util la scrierea programelor C care folosesc interfața API MySQL pentru C în vederea accesului la tabel. Dacă folosiți Tcl/Tk, este de asemenea disponibil un generator de ecran de interfață.
- **Pachete care extind caracteristicile sistemului MySQL.** Acestea includ instrumente pentru căutarea de text cu facilități complete și instrumente care vă ajută să reproduceți bazele de date MySQL.
- **Instrumente de conversie a datelor.** Acestea includ programe care facilitează conversia datelor folosite în alte programe în vederea utilizării în MySQL și invers. De exemplu, puteți converti fișierele `.dbf` în tabele MySQL și viceversa, respectiv tabele Access în tabele MySQL și invers.
- **Suport pentru ODBC.** Standardul ODBC (Open Database Connectivity) dezvoltat de Microsoft permite programelor compatibile ODBC să obțină acces la bazele de date MySQL, dacă este disponibil un driver MySQL. De exemplu, puteți instala MyODBC în Windows și apoi puteți folosi programe ODBC sub Windows pentru a obține acces la MySQL. O aplicație comună care implică sistemul MyODBC constă în conectarea la un server MySQL din Microsoft Access. Pentru aceasta, este necesar ca MyODBC să fie configurat sub Windows și trebuie să creați un nume al sursei de

date (DSN) pentru conectarea la MySQL. (Dacă nu ați configurat MyODBC sau nu ați creat un DSN, consultați Anexa A, „Obținerea și instalarea programelor".) După ce ODBC a fost configurat, folosiți următoarea procedură pentru a vă conecta la serverul MySQL din Access:

1. Porniți programul Access.
2. Creați o nouă bază de date vidă.
3. Din meniul File, selectați External Data, apoi Link Tables.
4. În fereastra care apare, executați clic pe meniul pop-up Files of type și selectați ODBC Databases.
5. Selectați numele DSN pe care l-ați specificat în panoul de control ODBC pentru a vă conecta la MySQL.
6. Începeți operația de legare.
7. Selectați tabelele pe care doriți să le vedeți.
8. Pentru a deschide un tabel, executați dublu clic pe numele acestuia.

- **Interfețe cu alte programe.** Numeroase instrumente au fost concepute pentru a vă ajuta să încorporați MySQL drept componentă back end pentru alte programe. Există remedii pentru Apache și WU-ftpd care determină aceste programe să execute consemnarea în tabelele MySQL. Puteți folosi MySQL pentru a executa autentificarea utilizatorilor de programe poștale pentru `qmail` și serverul IMAP Cyrus. Autentificarea cu programul `pam` se poate executa folosind de asemenea sistemul MySQL.
- **Instrumente administrative.** Programele care facilitează gestionarea instalării sistemului MySQL se prezintă sub diferite forme. Unele sunt programe în linie de comandă, iar multe altele sunt instrumente GUI. Unii clienți grafici se bazează pe X Windows System, un altul rulează sub Windows, iar alții, scrisi în Perl și PHP, sunt bazați pe Web.

ANEXA J

Furnizori de servicii Internet

Mulți oameni au acces complet[1] la Internet și la MySQL prin intermediul unei conexiuni rapide. Aceasta se întâmplă mai ales în medii universitare și în cadrul unor companii suficient de mari pentru a-și avea propriile lor departamente de prelucrare automată a datelor. Dar pentru multe alte persoane această situație nu este valabilă, iar accesul la serviciile on-line este asigurat de un furnizor de servicii Internet (Internet Service Provider - ISP). În afară de conectarea elementară la Internet pentru servicii precum poșta electronică și navigarea prin Web, mulți furnizori oferă și alte servicii, precum accesul la MySQL și un server Web.

Acest capitol reprezintă un ghid pentru selectarea unui furnizor de servicii Internet adecvat cerințelor dumneavoastră. Puteți avea nevoie atât de o modalitate de a vă conecta la Internet, cât și de o gazdă care furnizează sistemul MySQL. Alternativ, este posibil ca MySQL să ruleze deja pe calculatorul dumneavoastră și aveți nevoie de un furnizor la care să vă puteți conecta și care va permite accesul la calculatorul dumneavoastră a cererilor de conexiune sosite prin Internet.

O modalitate de a-i depista pe potențialii furnizori este de a vizita siturile Web pentru a căuta pachetele de care aveți nevoie, cu scopul de a vedea dacă acestea conțin o pagină despre furnizorii care găzduiesc serviciul. De exemplu, situl Web PHP are o pagină de căutare destinată găsirii furnizorilor de servicii Internet care oferă serviciul PHP. Unul dintre criteriile de căutare vă permite să căutați furnizorii care oferă MySQL. Există de asemenea un sit care enumeră furnizorii sistemului MySQL. Puteți vizita aceste situri la următoarele adrese:

```
http://hosts.php.net/search.php3
http://www.wix.com/mysql-hosting/
```

Nu toți furnizorii sunt la fel, deci este important să faceți o analiză comparativă înainte de a alege unul. Alegeți cu atenție; un furnizor de proastă calitate vă poate „lecui" de plăcerea utilizării sistemului MySQL, în timp ce unul bun vă poate ajuta enorm. Veți asuda puțin căutând, dar, dacă nu căutați, puteți face o alegere neinspirată și apoi va trebui oricum să investiți timpul respectiv.

Majoritatea criteriilor prezentate în această anexă trebuie evaluate pe o bază relativă. S-ar putea să nu găsiți un furnizor care să vă satisfacă integral pretențiile, dar trebuie să puteți găsi unul care satisface aceste cerințe mai bine decât alții.

[1] Adică 24 de ore pe zi. – N.T.

Când căutați, încercați să vă păstrați perspectiva. Nu toți furnizorii de servicii Internet sunt „băieți răi", care încearcă să vă jumulească de o sumă de bani cât mai mare și să vă ofere servicii de o valoare cât mai mică. Dar asta nu înseamnă că sunt „băieți buni" cu toții. Este important să vă faceți „temele" așa cum trebuie, astfel încât să găsiți unul dintre furnizorii care își cunosc marfa și care vă pot ajuta să vă atingeți scopurile. În timpul investigațiilor, fiți atent la furnizorul care evită să răspundă la întrebări concrete cu privire la practicile de lucru sau la caracteristicile tehnice. De asemenea, feriți-vă de furnizorii care dau răspunsuri generale atunci când dumneavoastră cereți detalii.

O notă pentru furnizorii de servicii: MySQL reprezintă un serviciu pe care îl puteți oferi clienților dumneavoastră, cu prețul unor costuri financiare mult mai reduse decât ale aproape oricărui alt sistem de baze de date. Desigur, puteți folosi MySQL și intern, pentru propriile dumneavoastră activități (de exemplu, pentru a vă ajuta să păstrați evidența înregistrărilor aferente clienților). Dacă intenționați să instalați sistemul MySQL, citiți capitolul „pe invers". Adică: acolo unde se spune „Întrebați furnizorul dacă așa și pe dincolo", întrebați-vă dacă puteți răspunde sau nu la întrebarea în cauză. Dacă nu puteți, de ce? Acolo unde se pune întrebarea „Asigură furnizorul serviciile cutare sau cutare?", puneți-vă întrebarea dacă oferiți sau nu serviciile respective. Dacă nu, ce este necesar pentru a le oferi?

Primii pași în căutarea unui ISP

Prima etapă în identificarea unui furnizor de servicii este estimarea propriilor dumneavoastră necesități. Înainte de a găsi un furnizor care vă satisface pretențiile, trebuie să știți care sunt cerințele dumneavoastră în ceea ce privește serviciile și lărgimea de bandă. Poate că am afirmat un truism; din păcate însă, unii caută un ISP pornind numai de la ideea vagă că „vor să se conecteze". Un deziderat ca acesta este greu de realizat într-o manieră satisfăcătoare; multe întrebări sunt lăsate fără răspuns, deoarece nici măcar nu au fost puse. Conectat pentru ce? Pentru a realiza ce fel de activități? Care este costul considerat rezonabil?

După ce știți ceea ce vreți, puteți aborda furnizorii de servicii Internet cu un set comun de întrebări și îi veți putea compara mai bine. Dacă nu determinați în prealabil ceea ce doriți să știți, o dată cu fiecare furnizor veți învăța care sunt întrebările ce trebuie puse, veți obține informații incomplete și va fi mai dificil să le comparați.

Ca o componentă a evaluării dumneavoastră, puteți da unele telefoane preliminare pentru a vedea dacă puteți găsi un ISP sau doi care oferă un serviciu de evaluare, pentru a vă ajuta să vă determinați necesitățile. Dacă serviciul este efectiv orientat spre a vă ajuta să vă specificați cerințele (fără a fi doar o altă tehnică de vânzări pentru a vă „vârî pe gât" servicii pe care nu le doriți sau de care nu aveți nevoie), atunci poate că ați găsit un furnizor care are cel puțin un avantaj: un serviciu de relații cu clienții.

În timpul căutărilor dumneavoastră, rețineți că unele instituții care furnizează servicii, precum accesul la MySQL și găzduirea unui sit de Web, nu furnizează un serviciu de conectare prin apel telefonic. Poate că în cazul dumneavoastră soluția cea mai bună constă în alegerea unui furnizor pentru conectarea elementară prin apel telefonic, care vă permite accesul la Internet, respectiv a altui furnizor pentru activități legate de baze de date și Web. (Sau poate că aveți deja un furnizor pentru accesul prin linie telefonică de

care sunteți mulțumit, caz în care trebuie să găsiți un furnizor numai pentru servicii de baze de date și Web.) Secțiunile următoare, intitulate „Lărgime de bandă" și „Servicii", descriu tipurile de probleme rezolvate în modul cel mai adecvat de fiecare tip de ISP.

Lărgime de bandă

Furnizorii oferă diverse opțiuni, de la linii dedicate (mai rapide) la conexiuni prin linie telefonică și modem (mai lente). În general, se face un compromis între cost și viteză: accesul mai rapid costă mai mult. Dacă veți deplasa o mare cantitate de date între calculatorul dumneavoastră și gazda furnizorului, o conexiune prin modem poate fi prea lentă. Totuși, în anumite situații, vă puteți descurca folosind o conexiune mai lentă între calculatorul dumneavoastră și ISP. De exemplu, dacă majoritatea activităților pe care le executați cu gazda furnizorului se datorează clienților care obțin accesul la baza dumneavoastră de date din amonte (prin intermediul unui sit Web, de exemplu), atunci majoritatea traficului dumneavoastră legat de bazele de date va fi orientat dinspre ISP spre exterior.

Servicii

Necesitatea cea mai evidentă pentru orice potențial ISP este aceea de a furniza MySQL, dar și următoarele servicii pot fi necesare sau de dorit:

- **Un cont de e-mail.** Acesta este esențial pentru comunicarea cu personalul tehnic de asistență al furnizorului și îl puteți folosi pentru înscrierea într-una sau mai multe dintre listele de corespondență conexe sistemului MySQL.
- **Acces prin Telnet la un cont de shell.** Acesta vă oferă posibilitatea de a rula utilitare UNIX standard și de a folosi clienții MySQL în linie de comandă precum `mysql`, `mysqldump` și `mysqlimport`. De asemenea, vă veți putea instala propriile dumneavoastră programe.
- **Servicii suplimentare conexe sistemului MySQL.** Acestea pot include găzduirea unui sit Web, PHP și funcționalități CGI.
- **FTP.** Acesta este util pentru transferul fișierelor între calculatorul dumneavoastră și sistemul furnizorului ISP. De exemplu, probabil că aveți unele fișiere de date pe care doriți să le încărcați în baza dumneavoastră de date pentru a o popula inițial sau puteți genera ieșiri de la baza de date pe care doriți să le manipulați pe propriul dumneavoastră calculator. De asemenea, doriți să permiteți altor persoane să descarce fișierele dumneavoastră pe sistemele proprii.
- **Înregistrarea numelor de domenii.** Prin intermediul acestei operații, primiți adrese de e-mail și sit Web sub propriul dumneavoastră nume de domeniu, și nu sub domeniul furnizorului dumneavoastră. Acesta este un serviciu oportun dacă doriți să creați o prezență Web sau FTP sub un nume care să poată fi identificat imediat cu instituția sau cu firma dumneavoastră.

Evaluarea unui ISP – considerații generale

Această secțiune enumeră câteva criterii de ordin general pe care le puteți folosi pentru a evalua potențialii furnizori de servicii Internet. Secțiunea următoare va aborda aspecte specifice sistemului MySQL. Când abordați un nou ISP, puneți-vă întrebările următoare:

- ✓ Cât este de simplă obținerea informațiilor?
- ✓ Care sunt costurile?
- ✓ Care sunt tipurile de programe pentru accesul clienților oferite de ISP?
- ✓ Ce fel de asistență inițială primiți din partea furnizorului?
- ✓ Dispune furnizorul de o conectivitate și de o lărgime de bandă adecvate?
- ✓ Care sunt opțiunile furnizorului pentru conectarea clienților?
- ✓ Ce fel de asistență tehnică oferă furnizorul?
- ✓ Există cote? Dacă da, care sunt acestea?
- ✓ Ce tip de echipamente oferă furnizorul?
- ✓ Cum rezolvă furnizorul problemele legate de confidențialitate și securitate?
- ✓ Ce fel de reputație are furnizorul și de când își desfășoară activitatea în domeniu?
- ✓ Care sunt planurile de dezvoltare ale firmei care furnizează serviciile?

Să discutăm în detaliu despre fiecare din aceste întrebări:

- ✓ **Cât este de simplă obținerea informațiilor?** Când sunați pentru a vă interesa cu privire la serviciile pe care le furnizează un ISP, răspunde acesta la telefon? Dacă aveți probleme în a obține o linie deschisă, poate că este vorba despre o activitate care se desfășoară serile, în garajul cuiva. Dacă vă răspunde un robot telefonic, veți primi un telefon de la furnizor?

  Operatorii serviciului de relații cu clienții cunosc răspunsurile la întrebările dumneavoastră? Dacă nu, pot găsi pe cineva din firmă care să le cunoască? Sau încearcă să vă convingă că problemele pe care le ridicați nu sunt importante (ceva de genul „Sigur că da, putem face asta, n-aveți grijă")? Dacă nu vă pot răspunde la întrebări înainte de a pune mâna pe banii dumneavoastră, cum vă vor trata după aceea?

  Puteți înțelege termenii folosiți de personalul furnizorului atunci când discută cu dumneavoastră? Dacă singurul limbaj pe care îl vorbesc este jargonul de marketing, atunci ați luat legătura cu o firmă care ține mai mult să-și vândă serviciile și care nu prea dispune de cunoștințe tehnice solide. Pe de altă parte, dacă vă sufocă de atâta jargon, atunci aveți de-a face cu o adunătură de „expertoizi" care își cunosc bine domeniul, dar care sunt incapabili să comunice adecvat cu oamenii obișnuiți.

  Vizitați situl Web al furnizorului, dacă aveți acces la un browser. Situl este clar, conține informații utile și este ușor de înțeles? Dacă furnizorul nu este în stare să încropească o pagină de bază performantă, atunci care va fi următorul său „rateu"?

- ✓ **Care sunt costurile?** Acestea pot varia considerabil, deoarece depind de un număr foarte mare de factori. Există o taxă inițială de abonament sau configurare în afară de taxa lunară normală? Taxa lunară este fixă sau variază? Ce include taxa? Aveți timp de

conectare nelimitat sau există o limită orară? Dacă depășiți limita, ce se întâmplă? Furnizorul refuză conectarea (ceea ce nu este prea convenabil, dar nu vă costă nimic în plus), sau există o taxă de depășire dincolo de taxa dumneavoastră lunară normală? Dacă există un tarif, este inclus și spațiul pe disc?

Pentru conexiunile prin apel telefonic, este de presupus că furnizorul are un număr local pe care îl puteți folosi în zona dumneavoastră de apelare. Dispune furnizorul de un număr de telefon special pe care îl puteți folosi atunci când nu vă aflați în zona respectivă? Există o taxă suplimentară pentru utilizarea acestui număr? Dacă locuiți într-o zonă rurală, probabil că opțiunile dumneavoastră sunt mai limitate și cel mai important este să vă asigurați că accesul la furnizor nu implică un apel cu taxă.

Deseori, furnizorii includ în taxa de bază o cantitate modestă de spațiu pe disc (câțiva megaocteți), cu opțiunea de a cumpăra o porțiune mai mare, dacă este necesar. Obțineți informații exacte referitoare la taxele pentru utilizarea discului, dacă vă așteptați la baze de date de mari dimensiuni sau dacă intenționați să găzduiți un sit Web (mai ales dacă va conține numeroase elemente grafice).

Dacă vă creați o prezență Web sau FTP pe gazda furnizorului, care sunt taxele percepute de acesta pentru volumul fișierelor descărcate atunci când utilizatorii obțin accesul la paginile dumneavoastră de Web sau când descarcă fișiere? Unii furnizori impun o cotă și, dacă depășiți limita, sunteți blocat până la începutul perioadei următoare. Alții pot include în nota dumneavoastră de plată o taxă variabilă, bazată pe volumul fișierelor descărcate. Iar alții vă pot schimba categoria de clasificare în „client comercial" (probabil la o taxă mai mare). Aflați care sunt opțiunile aflate la dispoziția dumneavoastră.

Asistența tehnică este gratuită sau există o taxă? Dacă este gratuită, este numai pentru o perioadă limitată de timp, de exemplu pentru primele 30 de zile? Dacă există o taxă, aceasta este fixă sau există o taxă per incident? Se poate achita o taxă suplimentară pentru a beneficia de asistență în condiții speciale?

Dacă vă este prezentat un plan de achitare în avans a taxei, evaluați-l ținând cont de istoricul și de reputația companiei. Opțiunile de plată în avans sunt uneori folosite de firmele aflate la început, ca mijloc de acumulare a unui capital de operare, dar asemenea furnizori au tendința de a dispărea de pe piață, caz în care taxele dumneavoastră plătite în avans sunt irecuperabile.

Trebuie să semnați un contract pe termen lung (de un an sau mai mult)? Dacă nu sunteți sigur de posibilitățile unui furnizor, doriți să aveți posibilitatea de a renunța la serviciile acestuia după o lună-două. E ușor să închei un abonament; cât de ușor este să anulați abonamentul, dacă serviciile firmei nu vă satisfac? Vă puteți recupera porțiunea nefolosită a contractului dumneavoastră?

Poate vă puteți procura un cont de testare gratuit sau la un preț redus, care vă va permite să estimați competența tehnică și serviciile de asistență ale furnizorului. Rețineți că, în mod caracteristic, un asemenea cont va include limitări ale activităților dumneavoastră. De exemplu, restricțiile privind spațiul pe disc vor limita dimensiunea oricărei baze de date de test pe care o puteți crea.

✓ **Care sunt tipurile de programe pentru accesul clienților oferite de ISP?** Furnizorul acceptă platforma dumneavoastră? Unii furnizori pot specifica faptul că acceptă numai anumite sisteme, cum este Windows, ceea ce nu vă este de folos dacă rulați UNIX sau Mac OS.[2]

Oferă furnizorul un client de e-mail și un browser Web, dacă aveți nevoie de aceste servicii? Trebuie să le folosiți pe cele oferite de furnizor sau puteți folosi și altele, dacă preferați?

✓ **Ce fel de asistență inițială primiți din partea furnizorului?** Oferă furnizorul asistență în ceea ce privește conectarea și punerea în funcțiune a contului dumneavoastră? Există o taxă pentru acest serviciu (ceea ce este posibil dacă aveți cerințe complexe)?

✓ **Dispune furnizorul de o conectivitate și de o lărgime de bandă adecvate?** Cât de aproape este furnizorul de coloana vertebrală a Internetului? Furnizorul este conectat direct sau se folosește de un alt furnizor pentru a ajunge acolo? Furnizorul dispune de conexiuni redundante în cazul apariției unei pene de conexiune?

Cât de mare este „conducta" dintre ISP și coloana vertebrală? Tipul de conexiune determină *lărgimea de bandă* a acesteia, adică volumul maxim de trafic pe care îl poate manipula furnizorul între coloana vertebrală și propriile sale sisteme. Liniile T-3 și T-1 sunt trunchiuri de lărgime de bandă mare, care asigură conexiuni rapide, dar, în cazul în care propriile conexiuni de ieșire ale furnizorului se realizează prin modeme, va apărea o strangulare și vitezele de transmisie vor avea de suferit.

Interesați-vă cu privire la traficul care trece prin conexiunea furnizorului – care este procentul de utilizare al capacității acestuia? Dacă împărțiți cu mulți alți utilizatori activi o conexiune la coloana vertebrală, lărgimea de bandă disponibilă pentru dumneavoastră scade, ceea ce nu vă va încetini numai pe dumneavoastră; o legătură saturată reduce viteza de acces a persoanelor care se conectează din amonte la baza de date sau la situl dumneavoastră de Web, dacă aveți unul.

✓ **Care sunt opțiunile furnizorului pentru conectarea clienților?** Dumneavoastră puteți dispune de un modem la 56Kbps, dar și furnizorul trebuie să-l accepte; în caz contrar, veți fi limitat la viteze de conexiune mai reduse. Oferă furnizorul opțiuni de viteză mai mare, precum ISDN sau DSL? Dacă vă așteptați să manipulați cantități mari de trafic între situl dumneavoastră și furnizor, este de dorit chiar să configurați o linie dedicată. Vă poate ajuta furnizorul în acest sens?

Nu cereți prea mult de la un ISP

Nu aveți cum să beneficiați de acces nelimitat, spațiu pe disc nelimitat și asistență tehnică nelimitată pentru 20 de dolari pe lună. Sunteți dispus să plătiți pentru un serviciu și o asistență de calitate? Doriți să primiți ceva în schimbul banilor dumneavoastră, dar și furnizorii se așteaptă la compensații rezonabile pentru serviciile pe care le oferă.

[2] Dacă vă conectați prin intermediul protocoalelor PPP (protocol punct la punct) sau SLIP (IP prin linie serială), probabil că vă puteți conecta de la practic orice platformă. – N.A.

Dacă vă veți conecta cu ajutorul modemului, nu doriți să primiți un semnal „ocupat" atunci când încercați să ajungeți la furnizorul de servicii. Vă puteți interesa la diverșii furnizori care este raportul dintre clienți și modeme, dar aceasta depinde de activitatea clienților. Clienții rezidențiali, care se conectează pentru scurtă vreme, nu ocupă rezerva de modeme în aceeași măsură ca și clienții comerciali, care ocupă toată ziua o conexiune pentru a-și dirija afacerile. Cel mai bine este să întrebați cât de frecvent veți primi un semnal de linie ocupată, mai ales la ora din zi la care vă veți conecta.

Puteți face următorul test înainte de a vă abona: formați numărul de telefon al furnizorului de la un telefon obișnuit și ascultați scârțâiturile modemului aflat la celălalt capăt al firului. Efectuați acest test de mai multe ori, în mai multe zile. Cât de des telefonul sună ocupat sau nu primiți nici un răspuns? Comparați disponibilitatea modemului cu activitățile dumneavoastră. Dacă sunteți o firmă, doriți să fiți sigur că vă puteți conecta ori de câte ori doriți. Pentru scopuri personale, poate că semnalul ocazional de linie ocupată nu vă deranjează.

✓ **Ce fel de asistență tehnică oferă furnizorul?** Asistența tehnică este disponibilă permanent sau numai în timpul orelor normale de serviciu? Poate că intenționați să vă conectați numai în timpul zilei; în caz contrar, se va găsi cineva care să vă răspundă la întrebare? Personalul de asistență tehnică dispune și de telefon, și de e-mail? Aveți nevoie de un telefon, deoarece nu puteți trimite mesaje e-mail dacă nu vă puteți conecta. Dar doriți să beneficiați și de e-mail, pentru a putea trimite personalului de asistență informații tehnice sau date de ieșire ale programului, pentru a evita încercarea de a le descrie la telefon. Vi s-a promis un răspuns într-un anumit interval de timp? (Este posibil ca problema să nu se rezolve în intervalul respectiv, dar cererile trebuie măcar confirmate rapid, pentru ca să știți că se lucrează la rezolvarea lor.)

Există asistență on-line? Este clară și la obiect sau derutantă și vagă? O puteți parcurge cu ușurință, pentru a găsi informațiile de care aveți nevoie? Se pot executa căutări?

Puteți evalua personal disponibilitatea asistenței tehnice în același mod în care testați numerele de telefon pentru acces – dați-le un telefon. Poate că vi se pare ciudat să faceți asta înainte de a avea un cont la furnizorul respectiv, dar poate fi instructiv să determinați dacă sunteți pus sau nu pe „apel în așteptare" și pentru cât timp. De asemenea, întrebați care este numărul celor care lucrează în departamentul respectiv. Mai bine interesați-vă care este raportul client-tehnician.

✓ **Există cote? Dacă da, care sunt acestea?** Există cote pentru spațiul pe disc sau pentru timpul de prelucrare? Există restricții de timp pentru scripturile executate de serverul Web? Nu este ilogic ca furnizorul să impună unele limite pentru activitățile clientului, pentru a preveni monopolizarea resurselor partajate de alți clienți, dar doriți să știți care sunt aceste limite.

✓ **Ce tip de echipamente oferă furnizorul?** Se pot rula servicii Internet elementare pe un PC vechi de clasă 386, dar această soluție nu este adecvată pentru numere mari de utilizatori, pentru a nu mai aminti nimic de încercarea de a rula servicii care necesită o putere mai mare de prelucrare a datelor, precum MySQL. Acesta nu acaparează resursele sistemului, așa cum procedează sistemele de baze de date de mari dimensiuni, dar are totuși nevoie de un ISP care să ruleze echipamente cu o oarecare cantitate de... mușchi. Care sunt categoriile de încărcări pe care sistemul le poate manipula fără probleme?

Unde se află serverul? În garajul sau în pivnița cuiva? Un furnizor situat într-un district comercial poate convinge mai ușor compania de telefoane să-i aloce linii de trunchi suplimentare, pe măsura creșterii cerințelor.

Cum rezolvă furnizorul problema defecțiunii echipamentelor? Există un plan de refacere, sau sunteți pur și simplu îmbrâncit afară din rețea până la repararea sau înlocuirea echipamentului? Care este procentajul real de funcționare al furnizorului în ultimele luni? Cine este responsabil cu incidentele care cauzează căderea sistemului? Se execută copii de siguranță ale sistemului de fișiere, și dacă da, cât de frecvent? Sau, după părerea furnizorului, aceasta este responsabilitatea dumneavoastră?

Există timpi de nefuncționare planificați (pentru operațiile planificate de întreținere sau executarea copiilor de siguranță, de exemplu)? Îi informează furnizorul pe clienți cu privire la momentul de început al acestor perioade? Dacă rulați un server Web, probabil preferați ca acesta să fie disponibil 24 de ore pe zi.

✓ **Cum rezolvă furnizorul problemele legate de confidențialitate și securitate?** Are furnizorul o politică legată de confidențialitatea fișierelor dumneavoastră? Ce măsuri ia acesta pentru a preveni compromiterea contului dumneavoastră?

✓ **Ce fel de reputație are furnizorul și de când își desfășoară activitatea în domeniu?** Care este reputația furnizorului? Ce alți clienți mai are și cât sunt de satisfăcuți aceștia? Informațiile verbale pot fi utile; întrebați-vă cunoștințele, pentru a afla dacă au auzit de potențialii furnizori. Puteți cere furnizorului referințe și poate fi util să solicitați păreri în lista de corespondență MySQL. Probabil că un furnizor vă va trimite la persoane care sunt satisfăcute de serviciile acestuia. În lista de corespondență, probabil că aveți auzi păreri din ambele tabere. Abonații de acolo sunt foarte doritori să relateze despre calitatea bună sau proastă a experienței lor cu un furnizor.

De cât timp activează furnizorul respectiv? Industria furnizorilor de servicii Internet înregistrează un număr fenomenal de mare de firme falimentare sau de scurtă durată, deoarece a porni este mai ușor – a te menține este mai dificil. Un furnizor cu o oarecare longevitate are mai multe șanse de a exista și în viitorul mai îndepărtat și poate fi capabil de a vă ajuta pe termen lung.

✓ **Care sunt planurile de dezvoltare ale firmei?** Care este mărimea furnizorului? Probabil că nu doriți să folosiți o firmă mică-micuță, dar chiar nici cele mai mari servicii nu sunt în mod obligatoriu mai bune. Companiile mai mici pot fi mai apropiate de necesitățile propriilor clienți.

Care este ritmul de dezvoltare a firmei? Care este numărul de clienți de care dispun în prezent? Care era acest număr acum o lună? Dar acum un an? Puteți dori să colaborați cu o companie prosperă, dar dezvoltarea exercită presiuni asupra resurselor privind lărgimea de bandă, accesul clienților și asistența tehnică. Care este politica firmei de rezolvare a problemelor legate de creșterea încărcării?

Evaluarea unui ISP – considerații specifice sistemului MySQL

Secțiunea anterioară v-a oferit o modalitate de evaluare a caracteristicilor generale ale unui ISP. Această secțiune discută despre criteriile pentru evaluarea serviciilor MySQL ale unui ISP. Când vedeți scris „MySQL", interpretați această mențiune ca incluzând toate celelalte servicii conexe de care aveți nevoie, precum Apache, PHP și DBI:

- ✓ Care sunt serviciile conexe sistemului MySQL pe care le furnizează un ISP?
- ✓ Care este versiunea sistemului MySQL instalată?
- ✓ Care este politica furnizorului în ceea ce privește actualizarea sistemului MySQL?
- ✓ Sistemul de operare al furnizorului prezintă probleme cunoscute în ceea ce privește sistemul MySQL?
- ✓ Vă puteți instala propriile dumneavoastră programe?
- ✓ Confidențialitatea și securitatea bazei dumneavoastră de date reprezintă o preocupare pentru furnizor?
- ✓ Vă ajută personalul furnizorului la începutul activității dumneavoastră?
- ✓ Care este categoria de asistență tehnică specifică sistemului MySQL asigurată de furnizor?
- ✓ Câți dintre clienții furnizorului folosesc MySQL?

Să discutăm în detaliu despre fiecare dintre aceste întrebări.

- ✓ **Care sunt serviciile conexe sistemului MySQL pe care le furnizează un ISP?** Furnizorul trebuie să aibă deja instalat sistemul MySQL și toate celelalte pachete conexe de care aveți nevoie. Excepția poate fi aceea că aveți deja un furnizor de care sunteți mulțumit din alte puncte de vedere, dar care nu dispune de MySQL. În acest caz, vă puteți interesa la furnizor dacă acesta dorește să-l instaleze pentru dumneavoastră. Va trebui să folosiți un fel de abordare de afaceri pentru aceasta, și va trebui probabil să fiți de acord că o asemenea instalare se va face acceptând realitatea că MySQL nu este un serviciu care beneficiază de nivelul de asistență aferent serviciului de e-mail sau găzduirii de pagini Web.

  Vizitați situl Web al furnizorului. Conține acesta informații (mai mult sau mai puțin detaliate) despre serviciile MySQL oferite? Există o înțelegere de natură tehnică a sistemului sau situl abundă de jargonul folosit în marketing? Situl este ușor de parcurs? Conține acesta o zonă cu răspunsuri la întrebări puse frecvent de clienți cu privire la utilizarea serviciilor proprii, astfel încât să puteți căuta singur informațiile?

- ✓ **Care este versiunea sistemului MySQL instalată?** Este o versiune recentă sau o variantă veche de-a binelea, de care nu s-a mai atins nimeni de la instalarea acesteia? Examinați anexa cu notele de trecere de la o versiune la alta din manualul de referință MySQL, pentru a determina caracteristicile adăugate după versiunea rulată de furnizor. Aveți nevoie de aceste facilități?
- ✓ **Care este politica furnizorului în ceea ce privește actualizarea sistemului MySQL?** Furnizorul trebuie să creeze un echilibru între nevoia de stabilitate și de comportare cunoscută a aplicațiilor MySQL existente cu dorința clienților de a beneficia de noile caracteristici ale variantelor recente. Aceasta nu este în mod obligatoriu o problemă ușor de rezolvat, deci nu tratați preocupările furnizorului vizavi de aceasta ca pe ceva banal. Dar doriți să știți dacă există măcar o posibilitate de trecere la o versiune mai recentă, eventual pe o gazdă separată, special destinată testelor. Dacă vă oferiți să serviți drept cobai pentru variantele mai recente, veți putea ajunge la o soluție de cooperare, deoarece veți asigura furnizorului dumneavoastră un valoros serviciu de testare.

- ✓ **Sistemul de operare al furnizorului prezintă probleme cunoscute în ceea ce privește sistemul MySQL?** Examinați capitolul de instalare din manualul de referință MySQL, pentru a vedea dacă există limitări care vă pot afecta. Deseori, există soluții ocolitoare pentru aceste probleme.
- ✓ **Vă puteți instala propriile dumneavoastră programe?** În caz afirmativ, vi se permite să vă extindeți funcționalitățile propriului dumneavoastră sistem MySQL cu programe create de terțe părți sau concepute chiar de dumneavoastră.
- ✓ **Confidențialitatea și securitatea bazei dumneavoastră de date reprezintă o preocupare pentru furnizor?** În afară de măsurile generale luate de furnizor pentru protecția datelor personale ale clienților, furnizorul ia măsuri pentru protecția datelor MySQL în particular? Aveți propriul dumneavoastră server MySQL? Aceasta ar fi un avantaj din punctul de vedere al clientului, deoarece puteți controla persoanele care au acces la conținutul bazei dumneavoastră de date. În ceea ce privește furnizorul, această opțiune incumbă o cantitate de muncă suplimentară.
- ✓ **Vă ajută personalul furnizorului la începutul activității dumneavoastră?** Nu vă așteptați ca furnizorul să vă învețe SQL și să vă arate cum să vă scrieți interogările, dar trebuie să vă puteți conecta la serverul MySQL. Vi se spune ce anume trebuie să știți pentru a stabili o conexiune? Primiți un script demonstrativ pe care îl puteți rula pentru a verifica dacă aveți acces la baza dumneavoastră de date?
- ✓ **Care este categoria de asistență tehnică specifică sistemului MySQL asigurată de furnizor?** Din când în când, oamenii trimit mesaje în lista de corespondență MySQL, care conțin problemele avute cu furnizorul și la încercarea de a determina sistemul MySQL să ruleze în mod corespunzător. Am observat că, în mod destul de frecvent, furnizorul indică clientul drept sursă a problemei. Ca în multe alte controverse, este posibil ca în multe cazuri să existe greșeli de ambele părți, dar, din descrierile situațiilor apărute, pare clar că în multe situații furnizorul are prea puține cunoștințe privind modul de manipulare a sistemului MySQL și rezolvă problemele „pe ghicite". Din acest motiv, este important să determinați în prealabil dacă un furnizor potențial dispune sau nu de un personal administrativ sau tehnic care să aibă cunoștințe decente de MySQL. Întrebați-i pe membrii acestui personal care este nivelul lor de experiență tehnică. Interesați-vă dacă dispun de un contract de asistență pentru MySQL. În caz afirmativ, înseamnă că tratează sistemul MySQL cu toată seriozitatea și că sunt pregătiți să vă ajute la remedierea problemelor care pot apărea.

  De exemplu, are furnizorul experiență în manipularea fiecărui pachet de care aveți nevoie, astfel încât să poată oferi o asistență efectivă, sau se limitează să vă trimită la lista de corespondență? Furnizorul trebuie să cunoască materialul prezentat în Partea a III-a a acestei cărți, intitulată „Administrarea sistemului MySQL".

Feriți-vă de un administrator absent

Fiți atent la furnizorii care instalează sistemul MySQL doar ca să-l poată adăuga pe lista lor de servicii și să atragă mai mulți clienți. Nu doriți să aveți de-a face cu administratorul care încearcă să se fofileze cu o cantitate minimă de cunoștințe legate de îndatoririle sale administrative.

Va trebui să comparați asistența tehnică oferită cu nivelul de control asupra resurselor dumneavoastră pe care vi-l oferă furnizorul. Dacă doriți un control complet asupra serverului dumneavoastră MySQL, iar furnizorul vi-l oferă, este logic din partea lui să presupună că sunteți capabil de a administra serverul. În acest caz, doriți ca furnizorul să vă permită să asigurați un script care va porni serverul la încărcarea sistemului de operare al calculatorului, astfel încât să nu fiți obligat să porniți manual serverul la fiecare repornire a calculatorului.

✓ **Câți dintre clienții furnizorului folosesc MySQL?** Dacă sistemul MySQL este folosit în mod activ de mai mulți clienți, sunt șanse mai mari ca furnizorul să asigure o asistență de calitate decât dacă MySQL este folosit numai de unul sau doi clienți.

Index

Simboluri

& (ampersand), operator ȘI la nivel de biți, 532
* (asterisc)
  caracter de înlocuire, 63
  operator aritmetic, 527-528
\ (caractere backslash), modificarea semnificației ghilimelelor, 101
$ (simbolul dolarului), variabile PHP, 378
&& (ampersand dublu), operator logic, 533
|| (simbol cu două canale), operator logic, 533
= (semnul egal), operator de comparație, 529
\ (caracterul escape), caracter de înlocuire la echivalențele cu un model, 535
! (semnul exclamării), operator logic, 533
>= (mai mare sau egal), operator de comparație, 530
> (mai mare), operator de comparație, 530
<< (operator de deplasare la stânga), operator la nivel de bit, 532
<= (mai mic sau egal), operator de comparație, 530
- (semnul minus), operator aritmetic, 527
<=> (operator de egalitate ce acceptă valoarea NULL), 529
< > (paranteze)
  funcții, 42
  gruparea operatorilor, 527
% (simbolul procentului)
  caracter de înlocuire, 149, 447
  caracter de înlocuire la echivalențele cu un model, 534
  operator aritmetic, 527
  prompt de interpretor, 39
| (simbol canal), operator SAU la nivel de bit, 532
+ (semnul plus), operator aritmetic, 527
-# opțiune de program (date de ieșire de depanare), 599
-? opțiune de program (asistență on-line), 598, 599
? (semnul întrebării), comandă, clientul mysql, 615
\# (rehash), comandă, clientul mysql, 615
>> (operator de deplasare la dreapta), operator la nivel de bit, 532
; (punct și virgulă)
  caracter de terminare a instrucțiunilor, 42
  scripturi Perl, 298
; (punct și virgulă), comandă, clientul mysql, 615
/ (slash), operator aritmetic, 527
_ (liniuță de subliniere), caracter de înlocuire la identificarea echivalențelor cu un model, 534
!= (operator de inegalitate), operator de comparație, 529

A

-a, opțiune de program, 605, 605, 613
ABS( ), funcție numerică, 539
absențe, tabel (proiect de evidență a rezultatelor școlare), 59
acces client, 477-478
acces de la distanță, 28, 36
  acordarea privilegiilor, 447
acces pentru citire (securitate), 470
accesul la bazele de date (tabele de acordare), 451-453. *Vezi și* acces de la distanță
acordarea privilegiilor
  la nivel de bază de date, 448-449, 484-485
  permiterea accesului, 451
  privilegii specifice de tabel, 449-450
  specifice coloanelor, 449-450
  super-utilizator, 484-485
ACOS( ), funcție numerică, 539
activarea caracteristicii de consemnare în jurnal, 453
actualizare
  înregistrări, 88-90
  privilegii, 484-485
  serverul MySQL, 467
  tabelul de acordare user, 438
actualizări relative, 191
adăugare
  înregistrări, 60
  rânduri în tabele, 60-61
--add-drop-table, opțiune de program, 628
--add-locks, opțiune de program, 628
add_new_event( ), funcție, 398
ADDATE( ), funcție, 550
addslashes( ), funcție, 392
administrare
  atribuții generale de administrare, 416-417
  conturi de utilizator, 444-445
  depanare, 418
  întreținere, 418
  securitate, 417
  serverul MySQL
    pornire-oprire, 438
    rulare cu un cont fără privilegii, 438-440
  utilizator rădăcină, configurare, 437
  vedere de ansamblu, 415
afișare
  conținutul tabelului, 62
  dimensiuni, 136-137
  informații dată/oră, 137
  unități monetare, 136
afișarea datelor binare, 272
alegere. *Vezi* selectare
aliasuri, 71, 585
  convenții de nume, sensibilitatea la diferența între majuscule și minuscule, 162
  interpretoare, 92-93
aliasuri de coloană, 71
alinierea coloanelor (date de ieșire HTML), 384-385
ALL, cuvânt cheie, 583
ALL, tip de privilegiu
--allow-keywords, opțiune de program, 628
ALTER, privilegiu, 475, 575
ALTER TABLE, instrucțiune, 90, 173-175, 424, 564-566
  crearea indexurilor, 170-172
  ștergerea indexurilor, 172-173
ampersand (&), operator ȘI pe biți, 532
ampersand dublu (&&), operator logic, 533
amplasarea catalogului de date, 420-421
amplasarea prestabilită a catalogului de date, 420
analiza opțiunilor din linie de comandă, 257-266
--analyze, opțiune de program, 605
AND, operator, 66
  logici, 533
  pe biți, 532
anularea efectelor utilitarului de optimizare MySQL, 205-206
anularea interogărilor cu mai multe linii, 42
anularea interogărilor pe mai multe rânduri, 42
API (interfețe de programare a aplicațiilor), 222-224
  C. *Vezi* API C

factori pentru selectarea, 228
mediu de execuție, 229
performanțe, 230-233
portabilitate, 235-236
timp de dezvoltare a programului, 233-235
Perl DBI. *Vezi* API pentru Perl DBI
PHP. *Vezi* API pentru PHP
API C, 222-224, 237-238, 634
factori de selectare
limbaje compilate sau interpretate, 230
mediu de execuție, 229
portabilitate, 235-236
timp de dezvoltare a programului, 233-235
funcții, 640
administrative, 651.652
construcția interogărilor, 646-647
depanare, 659-660
gestiunea conexiunii, 632.637
informații, 657-658
prelucrarea setului de rezultate, 648-657
scoase din uz, 660-662
semnalarea erorilor, 645
mysql.h, fișier antet, 634
originea, 224
prelucrarea interogărilor, 266-268
abordări alternative, 275-276
calcule cu seturi de rezultate, 265, 267
care nu returnează seturi de rezultate, 268-269
codarea datelor problematice, 286-288
comparație între mysql_store_result() și mysql_use_result(), 276-278
încărcarea datelor imagine, 288-289
metadatele setului de rezultate, 278-283
rutină de uz general pentru tratarea interogărilor, 272-274
set de rezultate returnat, 269-272
structura cu informații despre tabel, 289
probleme
pointeri de buffer neinițializați, 292
pointeri neinițializați ai variabilelor de manipulare a conexiunilor, 290-291
set de rezultate ratat, 291
valori NULL în coloane, 291-292
programe client
cerințe de sistem, 238-239
cod modular de conexiune, 246-253
compilare și legare, 239-240
conexiuni cu serverul, 240-242
parametri de conexiune la rulare, 253-266
prelucrarea interactivă a interogărilor, 283-284
verificarea erorilor, 242-246
tipuri de date, 635
macrocomenzi accesoriu, 639
non-scalare, 635-639
scalare, 635
API Perl DBI, 222-226, 663
atribute, 675
atribute dinamice, 675
atribute generale ale variabilelor de manipulare, 675
atributele variabilelor de manipulare a bazelor de date, 675
atributele variabilelor de manipulare a instrucțiunilor, 676-679
cerințe de sistem, 293-294
exemplul cu Liga istorică
automatizarea procesului de înștiințare privind plata cotizațiilor, 337-343
editarea intrărilor aferente membrilor, 344-349
generarea ca document HTML, 350-352
generarea catalogului, 330-337
identificarea intereselor comune, 349-350
factori de selectare
interpretori autonomi sau modulari, 231-233
limbaje compilate sau interpretate, 230
mediu de execuție, 229
portabilitate, 235-236
timp de dezvoltare a programului, 233-235
funcții utilitare, 674
instalare (sisteme UNIX), 509-510
metode, 665-667
func(), 672
metode de clasă, 665-667
metode de manipulare a bazelor de date, 667-669
metode de manipulare a instrucțiunilor, 669-672
metode generale de manipulare, 672-673
scrierea scripturilor, 663
scripturi, 294
asocierea parametrilor, 318-319
câmpuri de înlocuire, 316-319
cicluri de preluare a rândurilor, 305-308
citarea șirurilor, 314-316
comparație cu PHP, 379
depanare, 322
depanare cu parametri de afișare, 322-323
depanare cu urmărire, 324-325
exemplu, 295-300
metadatele setului de rezultate, 325-330
număr de rânduri returnate, 308
parametri de conexiune, 319-322
prelucrarea înregistrărilor fără seturi de rezultate, 304-305
rezultate cu un singur rând, 380-310
seturi de rezultate complete, 310-313
tratarea erorilor, 300-303
valori NULL, 313-314
scripturi bazate pe Web, 353-354
browser demonstrativ pentru baze de date, 362-366
browser pentru interese comune (exemplul cu Liga istorică), 370-371
browser pentru punctaje din proiectul pentru evidența rezultatelor școlare, 366-369
conectarea la serverul MySQL, 360-361
configurarea serverului Apache, 354-355
scripturi CGI.pm, 355-360
tipuri de date, 294-295
variabile de mediu, 679
API PHP, 31, 222-223, 227-228, 680
addslashes(), funcție, 392
factori de selectare
interpretori autonomi sau modulari, 231-233
limbaje compilate sau interpretate, 230
mediu de execuție, 229
portabilitate, 235-236
timp de dezvoltare a programului, 233-235
funcții, 681-682
construcția interogărilor, 685-687
de baze de date, 700-701
gestiunea conexiunii, 682-683
prelucrarea seturilor de rezultate, 687-670

raportarea erorilor, 683-684
funcții de preluare a rândurilor, 389-390
influențe lingvistice, 380
instalare
în UNIX, 510-512
în Windows, 515
interogări
citare, 392
emitere, 386-387
prelucrarea rezultatelor, 387-388, 390
raportarea erorilor, dezactivare, 391
require(), funcție, 382
scripturi
autonome, 375
caracteristici, 373
coloane, completare, 115
comparație cu scripturi DBI, 379
delimitatori, 373-374
eliminarea mesajelor de eroare, 377-378
etichete, 375
fișiere include, 380-384
pagini Web, generare, 373-374
scriere, 680-681
tratarea erorilor, 385
tablouri, 379
acces la, 390
tratarea erorilor, 391-392
variabile, 379
aplicare
privilegii pentru conturi de utilizator, 448-449
privilegii pentru mai multe conturi de utilizator, 449-450
valori NULL în coloane, 90
aplicarea conversiei de tip, 156
aplicații. *Vezi și* programe; scripturi; utilitare
baze de date, SGBDR (sisteme de gestiune a bazelor de date relaționale), 33
clienți, 36
integrarea cu MySQL, 220-221
multiutilizator, controlul concurenței, 449-450
programe de calcul tabelar, comparație cu bazele de date, 54-55
sisteme de gestiune a bazelor de date, 26
acces de la distanță, 27
demonstrative, 28-31
aplicații multiutilizator (controlul concurenței), 36
aplicații Web
integrarea cu MySQL, 220-222
interfața API pentru PHP, 227-228
argumente, separare, 145
argumente legătură, omitere din apelurile PHP, 377-378
arhitectură client-server, 36-37
conectare cu serverul, 39-41
AS nume, scrierea la ieșire a aliasurilor, 71
ASCII(), funcție șir, 543
ASIN(), funcție numerică, 539
asistență DBI (Perl), instalarea pe sistemele de operare Windows, 514-515
asistență on-line (opțiuni de program), 598, 599
asistență tehnică, selectare ISP, 710, 713-714
asocierea parametrilor (scripturi Perl), 318-319
aspecte legate de licență (MySQL), 15-16, 31
asterisc(*)
caracter de înlocuire, 63
operator aritmetic, 527
ATAN(), funcție numerică, 539
ATAN2(), funcție numerică, 539
atribuire
numere unice, 112
la înregistrări, 53
permisiuni în cataloage, 471-472
valori dată-oră în coloane, 126-127, 130-132
atribute, 130
TIMESTAMP, 128-129
YEAR, 130
valori legale în coloane SET, 125
valori numerice în coloane, 107
valori șir în coloane, 117-120
BLOB, 120
CHAR, 118-120
ENUM, 121-125
TEXT, 120
VARCHAR, 118-120
atribute
coloane, 104
dată-oră, 130
coloane numerice
AUTO-INCREMENT, 58, 112-113, 569-570
DEFAULT, 111
NULL, 111
UNSIGNED, 111
ZEROFILL, 110
coloane șir, 126
Perl DBI, 674
atribute dinamice, 675
atribute generale ale variabilelor de manipulare, 675
atributele variabilelor de manipulare a bazelor de date, 675
atributele variabilelor de manipulare a instrucțiunilor, 676-679
ZEROFILL, 115
atribute ale variabilelor de manipulare a instrucțiunilor (Perl DBI), 676-679
atribute ale variabilelor de manipulare pentru baze de date (Perl DBI), 667-669
atribute dinamice (Perl DBI), 675
atribute generale ale variabilelor de manipulare (Perl DBI), 675
atribute scoase din uz specifice MySQL (Perl DBI), 677
atribute specifice de tip (coloane), 104
atribuții generale ale administratorului, 416-417
AUTO_INCREMENT, atribut, 58, 112-113, 569-570
coloane numerice, 110
compatibilitate înapoi, 113
automatizare
rotația fișierelor jurnal, 454
scripturi de verificare a tabelelor, 498
verificarea tabelelor, 496
available_drivers(), metodă (Perl DBI), 665
AVG(), funcție, 80-81, 558
AVG_ROW_LENGTH, cuvânt cheie, 570

B

-b, opțiune de program 610, 612, 616, 622
back_log, variabilă (serverul MySQL), 463
backslash-uri (\), modificarea semnificației caracterelor de citare, 101
--backup, opțiune de program, 610
--basedir, opțiune de program 612
bash (aliasuri), 92-93
--batch, opțiune de program, 612
baza de date demonstrativă
creare, 43-44
Liga istorică americană
tabele, 44-50
tabelul cu membri, 46-47
tabelul cu președinți, 45-46
obținere, 39
proiectul de evidență a rezultatelor școlare, 32
tabele, 50, 57
tabelul cu absențe, 59
tabelul cu elevi, 55-56
tabelul cu evenimente, 52-55, 58

tabelul cu punctaje, 50-51, 59
bază de date prestabilită
creare, 43
selectare, 163
baze de date
acces la (tabele de acordare), 473-475
comparație cu programele de calcul tabelar, 54-55
convenții de nume, 43, 161, 425-426
creare, 43-44
CREATE DATABASE, instrucțiune, 163, 566
DROP DATABASE, instrucțiune, 163, 572
executarea copiilor de siguranță, 457-458
încărcarea datelor, optimizare, 209-211
întreținere preventivă, planificare, 496
modificarea amplasamentului, 433-434
optimizarea interogărilor. *Vezi* optimizarea interogărilor
performanțele sistemului, 426-428
privilegii, 475
refacerea de la copiile de siguranță, 461-462
regăsirea informațiilor despre, 176
replicare, 461
reprezentare în catalogul de date, 423
SELECT, instrucțiune, 177
uniri complete, 178-179
uniri la stânga, 180-181
uniri normale, 178
selectare (script PHP), 378
SGBDR (sisteme de gestiune a bazelor de date relaționale), 33
SHOW DATABASES, instrucțiune, 50, 589
tabele
blocare, 494-495
coloane, 103
creare, 44-50, 57
DESCRIBE, instrucțiune, 49
deteriorate, detecție, 488-489
executarea copiilor de siguranță, 459
legare, 53
modificarea amplasamentului, 434
planificare, 44-47
proiect de evidență a rezultatelor școlare, 50-60
protocol de blocare internă, 492-494
reparare, 491
reprezentare în catalogul de date, 424
verificare, 490
transferul în fișiere text, executare, 458
USE, instrucțiune, 163, 595
baze de date relaționale, comparație cu programele de calcul tabelar, 54-55
BENCHMARK(), funcție, 559
BETWEEN...AND, operator de comparație, 530
biblioteca GNU Readline, 93-94
biblioteci client (API C), 237-238, 634
cerințe de sistem pentru programe client, 238-239
cod de conexiune modular (program client), 246-253
compilarea și legarea programelor client, 239-240
conexiuni server (programul client), 240-242
funcții, 640
funcții administrative, 658-659
funcții de construcție a interogărilor, 646-647
funcții de depanare, 659-660
funcții de gestiune a conexiunilor, 640-645
funcții de informare, 657-658
funcții de prelucrare a setului de rezultate, 648-657
funcții de raportare a erorilor, 645
funcții scoase din uz, 660-662
macrocomenzi accesoriu, 639
mysql.h, fișier antet, 634
parametri de conexiune la rulare (programul client), 253-266
pointeri de buffer neinițializați, 292
pointeri neinițializați ai variabilelor de tratare a conexiunilor, 290-291
prelucrarea interactivă a interogărilor (programe client), 283-284
prelucrarea interogărilor, 266-283
set de rezultate ratat, 291
tipuri de date, 635
tipuri de date non-scalare, 635-639
tipuri de date scalare, 635
valori NULL în coloane, 291-292
verificarea erorilor (program client), 242-246
--big-tables, opțiune de program, 622
BIGINT, tip de date, 103, 107, 518
BIN(), funcție șir, 543
binare, precompilate, 502
binare precompilate, 501
BINARY, atribut (coloane șir), 126
BINARY, operator de conversie, 533
--bind-address, opțiune de program 612
bind_col(), metodă (Perl DBI), 669
bind_columns(), metodă (Perl DBI), 670
bind_param(), metodă (Perl DBI), 670
BIT_AND(), funcție de sumar, 558
BIT_COUNT(), funcție de sumar, 560
BIT_OR(), funcție de sumar, 558
BLOB (obiecte binare mari), 104
BLOB, tip de date, 103, 120, 521
blocare
GET_LOCK(), funcție, 561
privilegii la nivel de înregistrare, 194
RELEASE_LOCK(), funcție, 562
tabele, 494-495
blocare externă, 492-494
blocare internă, 492-494
blocări la citire, 493
blocări la scriere, 494
--block-search, opțiune de program, 606
--bootstrap, opțiune de program, 622
--brief, opțiune de program, 616
browser pentru interese comune (exemplul cu Liga istorică), 370-371
browsere
browser demonstrativ pentru baze de date, 362-366
browser pentru interese comune din Liga istorică, 370-371
browser pentru punctaje din proiectul pentru evidența rezultatelor școlare, 366-369

C

\c, comandă (clientul mysql), 614
-C, opțiune de program 589, 607, 629
calculul rezultatelor interogărilor, 71, 284-286
canal, simbol (||), operator SAU pe biți, 533
canal dublu, simbol (||), operator logic, 533

caracter punct și virgulă (;)
caracter de terminare a instrucțiunilor, 41
scripturi Perl, 298
caractere, valori NULL, 48
caractere de înlocuire, 149
asterisc (*), 63
clauze LIKE, 204
operatori de identificare a echivalențelor cu un model, 534
simbolul procentului, 447
caractere escape, 101
caracter de înlocuire pentru identificarea echivalenței cu un model, 534
caractere speciale, 579
fișiere cu opțiuni, 603
modulul CGI.pm, 357
Perl DBI, 299
șiruri, 535
caractere speciale (secvențe escape), 579
caracteristică multiutilizator, 27
caracteristici neacceptate ale sistemului MySQL, 189-194
caracterizarea domeniilor de valori, 139
cataloage
exemplul cu Liga istorică, generare, 330-337
fișiere de pornire, 440-441
nesigure, localizare, 470-471
permisiuni, atribuire, 471-472
catalog de date, 419
accesul serverului la, 422-423
administrare MySQL, 416
amplasare, 420-421
baze de date
modificarea amplasamentului, 433-434
reprezentare, 423
convenții de nume, 425-426
fișiere de stare, 428-431
modificarea amplasamentului, 435
modificarea amplasamentului, 431-435
evaluarea efectului, 432-433
metode de, 431-432
performanțele sistemului, 426-428
securitate, 430
sisteme UNIX, inițializare, 507
structură, 421-422
tabele
modificarea amplasamentului, 434
reprezentare, 424
câmpuri de înlocuire (scripturi Perl), 316-319
CEILING(), funcție numerică, 540
cereri de căutare în inventar bazate pe Web, 28
cerințe
instalare, 38
instrumente, 501
interfața API Perl DBI, 293-294
programe client, 239
repararea bazelor de date, 488-489
stocare
șiruri, 117
tipuri de coloane numerice, 106-107
valori dată-oră, 126-127
stocarea tipurilor de coloane numerice, 106-107
cerințe de sistem
interfața API pentru Perl DBI, 293-294
programe client, 239
cerințe privind spațiul de stocare
coloane
tipuri de coloane numerice, 107
valori dată-oră, 126-127
valori șir, 117
CGI.pm, modul, 354-356
caractere escape, 357
parametri de intrare, 356-357
transferul parametrilor din paginile Web, 358-360
CHANGE, clauză (instrucțiunea ALTER TABLE), 174
CHAR, tip de date, 103, 520
combinare cu VARCHAR, 118-120
CHAR(), funcție șir, 543
CHARACTER SET, cuvânt cheie, 586
CHARACTER_LENGTH(), funcție șir, 543
CHAR_LENGTH(), funcție, 543
CHECK, clauză (instrucțiunea CREATE TABLE), 164
--check, opțiune de program, 607
check_response(), funcție, 404
CHECKSUM, cuvânt cheie, 570
chei externe, 192-193
cheie primară, generare numere unice, 111
ChopBlanks, atribut (Perl DBI), 675
--chroot, opțiune de program, 622
cicluri de preluare a rândurilor, scriere cu scripturi Perl, 305-308
citare (interogări PHP), 392
citarea șirurilor în scripturi Perl, 314-316
clear, comandă (clientul mysql), 614
COALESCE(), funcție șir, 543
coloană AUTO_INCREMENT
inserție în tabele, 186
reluarea secvenței, 187
coloană cu secvență de numere
inserție în tabele, 186
reluarea secvenței, 187
coloane, 103-104
aliasuri, 71
atribute, 104
AUTO_INCREMENT, atribut, 58
inserție în tabele, 186
caracteristici, 103
coloane cu privilegii (tabele de acordare), 473-474
coloane pentru domeniul de valabilitate, 478-479
Column name, 479
Db, 479
Host, 478
ordine de identificare a echivalențelor, 482
Password, 479
tabele de acordare, 473-474
Table name, 479
User, 479
combinare CHAR și VARCHAR, 118-120
completare (scripturi), 115
completarea referințelor la, 179
conversie de tip, 144
dată-oră, 104, 126-127
afișare, 137
atribute, 130
formatare, 130-132
TIMESTAMP, 128-129
valori ambigue ale anilor, 132
YEAR, 130
date de ieșire în format HTML, aliniere, 384-385
date duplicate, 53
declarare (sintaxă), 104
DEFAULT, 105
denumire, 61
sensibilitate la diferența între majuscule și minuscule, 161-162
ENUM, 121-125
sortare, 125
valori corecte, 122
enumerare, 57, 103
formatare, 48
identificarea echivalențelor cu un model, 103
întregi, 57

lățime, declarare, 107
modificarea tipurilor, 174
nume complet determinate, 161
numere unice, generare, 58
numerice, 103-104, 106
atribuire, 107
atributul AUTO_INCREMENT, 110
atributul NULL, 111
atributul UNSIGNED, 111
atributul ZEROFILL, 110, 115
cerințe de spațiu de stocare, 106-107
conversie de tip, 152
domenii, 106-107
ordini de sortare, specificare, 187
reluarea secvenței, 187
secvențe
generare, 116
valoare inițială, specificare, 115
selectare
înregistrări din, 63
pentru indexuri, 199-201
pentru interogări optimizate, 206-208
SET, 121-125
SHOW COLUMNS, instrucțiune, 589
sortare, 69
șir, 103-104, 117-120
atributul BINARY, 126
atributul DEFAULT, 126
atributul NULL, 126
cerințe privind spațiul de stocare, 117
combinare CHAR și VARCHAR, 118-120
TEXT, 120
valori BLOB, 120
TEXT, indexare, 120
tipuri, selectare, 133-138
aspecte de inter-relaționare, 143
aspecte de performanță, 139-140
compararea coloanelor de lungimi fixe și variabile, 140-141
compararea dimensiunii tipurilor, 140
domenii de valori, 138-139
efectul asupra comparației valorilor, 142
posibilitate de indexare, 142-144
valori NULL, 141
valori BLOB, indexare, 120
valori NULL; aplicare, 90
valori trunchiate, 109
valori, specificare, 60
coloane cu lungime variabilă, combinare cu coloane de lungime fixă, 119-120
coloane cu privilegii (tabele de acordare), 473-474
coloane de determinare a domeniului de valabilitate
conținut, 478-479
ordine de identificare a echivalențelor cu un model, 482
tabele de acordare, 473-474
coloane de enumerare, 57-58, 103
coloane de lungime fixă, comparație cu coloane de lungime variabilă, 119-120
coloane de tip întreg, 57, 101, 103, 106-107
AUTO_INCREMENT, atribut, 110, 114
NULL, atribut, 111
UNSIGNED, atribut, 106-107, 111
Column name, coloană (coloane pentru domeniul de valabilitate), 479
columns_priv, tabel (tabel de acordare), 474
combinare
coloane CHAR și VARCHAR, 118-120
instrucțiuni condiționale (operatorul OR), 66
privilegii, 481
comentarii
scripturi Perl, 297
sintaxă în codul SQL, 181, 194, 595-596
comentarii în stil C inserate în programe SQL, 181, 595
comenzi. *Vezi și* **instrucțiuni (SQL)**
clientul mysql, 615
editarea comenzilor (fișierul istoric al clientului mysql), 611
find, 421, 455
invocare, 438
kill, 444
ls, 462
mysqladmin, 420-421, 618-620
mysqld, invocare, 440
mysqldump, 176
mysqlshow, 176
Perl, 297
pornire (sisteme Solaris), 441
ps, 421, 444
reluare, 92
COMMENT, cuvânt cheie, 570
--commit, opțiune de program, 616
comparații
baze de date și programe de calcul tabelar, 54-55
coloane SET și ENUM, 124
copii de siguranță ale sistemului de fișiere și fișiere copie directă, 455-456
funcții
mysql_pconnect() și mysql_connect(), 377-378
require() și include(), 382
limbaje compilate și interpretate, 230
mysql și MySQL, 38
scripturi, PHP și DBI, 379
valori, efect asupra selecției tipului de coloană, 142
comparații între șiruri, sensibilitatea la diferența între majuscule și minuscule, 529
compatibilitate înapoi
atributul AUTO_INCREMENT, 112-113
tabel de acordare, 483-484
compilare
biblioteca client API C, 634
programe client, 239-240
completare
coloane (scripturi), 115
valori numerice, 110
--complete-insert, opțiune de program, 629
componente hardware
optimizarea interogărilor, 215
selectarea ISP, 710-711
Comprehensive Perl Archive Network (CPAN), 235
--compress, opțiune de program, 460, 599
CONCAT(), funcție, 152, 543
conectare
la server, 437
operatori logici (operator SAU), 66
conexiuni
cu serverul MySQL prin intermediul scripturilor bazate pe Web, 360-361
parametri, 91
fișiere cu opțiuni, 91-92
specificare, 44
servere
inițiere, 39-41
închidere, 41
parametri de conectare la rulare (program client), 253-266
program client modular pentru, 246-253
programe client pentru, 240-242

conexiuni cu serverul, program client pentru, 240-242
parametri de conectare la rulare (program client), 253-266
program client modular pentru, 246-253
conexiuni persistente (API PHP), 683
confidențialitate, selectarea ISP, 711, 713
configurare
mai multe servere, 466
pornire
opțiuni, 442
verificarea tabelelor, 442
serverul MySQL
pornire-oprire, 438
variabile, 463
utilizator rădăcină, 437
configurare, utilitar de, 502
configurații stocate, ștergere din utilitarul de configurare, 502
connect, comandă (clientul mysql), 615
connect(), metodă, (Perl DBI), 297-298, 665-666
tratarea erorilor, 300
consemnare
nume de utilizator, 39
parole, 39-40
secvența de pornire, 40
consemnare în jurnal, 453
--console, opțiune de program, 625
CONSTRAINT, clauză (instrucțiunea CREATE TABLE), 164
control
accesul clienților, 477-478
raportarea erorilor (PHP), 391
controlul concurenței, 36
conturi. *Vezi* **conturi de utilizator**
conturi de utilizator
acces, restricționare, 447
anonime, revocarea privilegiilor, 452
creare (sisteme UNIX), 504
fără privilegii, rularea serverului MySQL, 438-440
gestionare, 444-445
privilegii
revocare, 451
selectare, 448-449
specifice coloanelor, acordare, 449-450
securitatea serverului, 417
securitatea sistemului de fișiere, 417
super-utilizatori, creare, 448-449
conturi de utilizator fără privilegii, rularea serverului MySQL, 438-440
conținut (tabele de acordare), 473-475
CONV(), funcție șir
convenții de nume
baze de date, 43, 161, 425-426
coloane, 61, 104
instrucțiuni SQL, sensibilitate la diferența între majuscule și minuscule, 162
tabele, 425-426
sensibilitatea la diferența între majuscule și minuscule, 64
variabile de manipulare Perl DBI, 294-295
variabile Perl DBI, 664
conversie
la MySQL, 14-15
tipuri de coloane, 144
conversie de tip, 152-155
forțare, 156
testare, 155-156
valori incorecte, 157
copiere
interogări, 95
tabele, 167-168
tabele ISAM, 456
copii de siguranță
copie directă, 460
date sistem, 453
efectuare, 457-458
importanță, 416
informații șterse, 453
refacerea datelor, 461-462
replicarea bazelor de date, 461
rotația fișierelor jurnal, 453
transferul în fișiere text, 457-458
copii de siguranță de sistem, 457-458
copiere directă, 460
executare, 453
refacerea datelor, 461-462
replicarea bazelor de date, 461
rotația fișierelor jurnal, 453
transferul în fișiere text, 457-458
copii de siguranță prin copiere directă, 460
--copy, opțiune de program, 616
cost
aspecte privind acordarea licenței pentru sistemul MySQL, 15-16
selectarea ISP, 707-709
COS(), funcție numerică, 540
COT(), funcție numerică, 540
cote, selectare ISP, 710
cote de spațiu pe disc, selectare ISP, 710
COUNT(), funcție, 77-79, 558
CPAN (Comprehensive Perl Archive Network), 235
creare
baza de date demonstrativă, 43-44
conturi de utilizator (sisteme UNIX), 504
fișiere cu opțiuni, 91
pagini Web din scripturi PHP, 373-374
scripturi de interpretor, 92-93
super-utilizatori, 448-449
tabele, 44-47, 104
bază de date demonstrativă, 47-49
proiect de evidență a rezultatelor școlare, 57
crearea condițională a tabelelor (instrucțiunile CREATE TABLE), 166
create, comandă (utilitarul mysqladmin), 619
CREATE, privilegiu, 476, 575
CREATE DATABASE, instrucțiune, 43, 163, 423, 566
CREATE FUNCTION, instrucțiune, 567
CREATE INDEX, instrucțiune, 170-172, 434, 567
CREATE TABLE, instrucțiune, 47, 57, 105, 164-165, 424, 567-571
crearea condiționată a tabelelor, 166
rezultatele interogării SELECT, crearea tabelelor pornind de la, 167-169
tabele temporare, 167
criptare (funcția PASSWORD()), 479
criterii, regăsire
expresii, 71
specificare, 64-67
valoarea NULL, 67-68
crontab, fișier
editare, 454
script de verificare a tabelelor, executare, 496-497
CROSS JOIN, cuvânt cheie, 584
CRYPT(), funcție, 479
CURDATE(), funcție, 550
CURRENT_DATE, funcție, 550
CURRENT_TIME, funcție, 550
CURRENT_TIMESTAMP, funcție, 550
cuvinte-cheie
ALL, 583
AUTO_INCREMENT, 58, 569-570
AVG_ROW_LENGTH, 570
CHARACTER SET, 586

CHECKSUM, 570
CROSS JOIN, 584
DEFAULT, 569
DELAY_KEY_WRITE, 570
DELAYED, 213, 577
DISTINCT, 76, 583
DISTINCTROW, 583
HIGH_PRIORITY, 213, 583
HOSTS, 574
IGNORE, 571, 577
INNER JOIN, 584
INSERT_ID, 586
INTO, 577
JOIN, 584
LAST_INSERT_ID, 586
LEFT JOIN, 85, 584
LEFT OUTER JOIN, 181
LOCAL, 62, 578
LOGS, 574
LOW_PRIORITY, 212, 572, 576-577, 595
MAX_ROWS, 570
MIN_ROWS, 570
NATURAL LEFT JOIN, 181, 585
NOT NULL, 569
NULL, 569
PACK_KEYS, 570
PASSWORD, 570, 586
PRIMARY KEY, 569
PRIVILEGES, 574
READ, 581
REPLACE, 571
sensibilitatea la diferența între majuscule și minuscule, 161-162
SQL_AUTO_IS_NULL, 587
SQL_BIG_RESULT, 583
SQL_BIG_SELECTS, 587
SQL_BIG_TABLES, 587
SQL_LOG_OFF, 587
SQL_LOG_UPDATE, 587
SQL_LOW_PRIORITY_UPDATES, 587
SQL_SELECT_LIMIT, 588
SQL_SMALL_RESULT, 583
SQL_WARNINGS, 588
STATUS, 574
STRAIGHT_JOIN, 179, 584
TABLES, 574
TEMPORARY, 567
TIMESTAMP, 588
TYPE, 571
UNIQUE, 569
WRITE, 581

D

-d, opțiune de program, 629, 631
--datadir, opțiune de program, 622
--data-file-length, opțiune de program, 607
DATABASE(), funcție, 560
dată-oră
afișare, 137
coloane, 103-104
funcții, 550-557
tipuri de date, 103, 126-127, 523
atribute, 130
DATE, 523
formatare, 130-132
TIME, 523
TIMESTAMP, 128-129, 524
valori ambigue ale anilor, 132
YEAR, 130, 524
valori, reguli de interpretare, 155
date, 72
regăsire, 72-74
TO_DAYS(), funcție, 73
DATE, tip de date, 104, 127, 523
date binare
afișare, 272
încărcare în tabele, 288-289
date de ieșire, formatare, 219-220
date duplicate, 53
date imagine, încărcare în tabele, 288-289
DATE_ADD(), funcție, 74, 550-552
DATE_FORMAT(), funcție, 552-553
DATE_SUB(), funcție, 74, 553
DATETIME, tip de date, 104, 127, 524
DAYNAME(), funcție, 553
DAYOFMONTH(), funcție, 72-73, 553
DAYOFWEEK(), funcție, 554
DAYOFYEAR(), funcție, 554
--db, opțiune de program, 617
Db, coloană (coloane cu domeniul de valabilitate), 479
db, tabel (tabel de acordare), 474
DBD (driver de baze de date) nivel, API Perl DBI, 224-225
DBI (interfață cu baza de date), nivel, 31
API Perl DBI, 224-225
DBI_TRACE, variabilă de mediu (Perl DBI), 324-325
--debug, opțiune de program 589, 617
--debug-info, opțiune de program, 612
DECIMAL, tip de date, 103, 106-107, 519
declanșatori, 193-194
declararea coloanelor
coloane ENUM, valori legale, 122
lățime, 107
sintaxă, 104
DECODE(), funcție, 560
--defaults-file, opțiune de program, 599
--default-table-type, opțiune, 622
DEFAULT, atribut, 105, 111
coloane șir, 126
DEFAULT, cuvânt cheie, 569
defined(), metodă (Perl DBI), 313-314
DEGREES(), funcție numerică, 540
DELAY KEY WRITE, 570
--delayed, opțiune (mysqldump), 459
--delayed-insert, opțiune de program, 629
DELAYED, cuvânt cheie, 213, 577
delayed-queue-size, parametru, 214, 463
--delete, opțiune de program, 631
DELETE, instrucțiune, 88, 571-572
atributul AUTO_INCREMENT, 114
clauza WHERE, 88-89
eliminarea conturilor de utilizator, 451
DELETE, privilegiu, 476, 575
delimitatori (scripturi PHP), 373-374
depanare
API C
pointeri de buffer neinițializați, 292
pointeri neinițializați la variabilele de tratare a conexiunilor, 290-291
set de rezultate ratat, 291
valori NULL în coloane, 291-292
atribuții administrative, 418
nume complet determinate (rezoluție de nume), 448-449
repararea tabelelor prin metoda standard, 492
depanarea scripturilor Perl, 322
cu instrucțiuni de afișare, 322-323
cu urmărire, 324-325
DESCRIBE, instrucțiune, 49-50, 176, 289, 572
trunchierea valorilor, 109
--description, opțiune de program, 606

despachetarea distribuțiilor, 502
distribuția samp_db, 515
distribuții sursă, 502
detectarea tabelelor deteriorate, 488-489
determinare
interogări, 83
referințe de coloană, 179
dezactivarea facilității de raportare a erorilor (PHP), 391
die(), funcție, 384-385
dimensiuni, afișare, 136-137
disconnect(), metodă (Perl DBI), 668
display_cell(), funcție, 397
display_entry(), funcție, 410-411
display_scores(), funcție, 398
DISTINCT, cuvânt cheie, 76, 583
DISTINCTROW, cuvânt cheie, 583
distribuția Msql-Mysql-modules, 225
distribuție
despachetare, 502
samp_db, instalare, 515
Windows, 512
distribuție binară (UNIX), instalare, 504-507
distribuții client-server (Windows), instalare, 513-514
distribuții sursă, despachetare, 502
do(), metodă (Perl DBI), 304-305, 668
documente HTML (exemplul cu Liga istorică), generare, 350-352
domeniu de valori
caracterizare, 139
selectarea tipurilor de coloane, 138-139
tipuri de coloane numerice, 106-107
DOUBLE, tip de date, 103, 106-107, 519
drop, comandă (utilitarul mysqladmin), 619
DROP, privilegiu, 476, 575
DROP DATABASE, instrucțiune, 163, 423, 572
DROP FUNCTION, instrucțiune, 572
DROP INDEX, instrucțiune, 172-173, 424, 573
DROP TABLE, instrucțiune, 169, 424, 573
dump_results(), metodă (Perl DBI), 670

E

\e, comandă (clientul mysql), 615
-e, opțiune de program, 606, 612, 614, 629
echi-uniri, 178
edit, comandă (clientul mysql), 615
editare
fișier crontab, 455
intrări aferente membrilor (exemplul cu Liga istorică), 344-349
editarea comenzilor (mysql), 93-94
fișier istoric pentru clienți, 611
editarea liniilor de intrare, 95
editarea on-line a intrărilor (Liga istorică americană), 405-412
eficiență, efectul asupra selecției tipului de coloană, 139-140
ego, comandă (clientul mysql), 615
elev, tabel (proiectul de evidență a rezultatelor școlare), 55-56
eliberarea memoriei alocate rezultatelor, 378-379
eliminare
baze de date mutate, 434
înregistrări, 88
înregistrări duplicate, 76-77
eliminarea mesajelor de eroare, 391
în scripturi PHP, 377-378
ELT(), funcție șir, 544
e-mail, selectare ISP, 706
emiterea interogărilor, 41
PHP, 386-387
privilegii de acces, 480-481
prompturi, 43
--enable-locking, opțiune de program, 622
ENCLOSE BY, clauză (instrucțiunea LOAD DATA), 579
ENCODE(), funcție, 560
ENCRYPT(), funcție, 560
enter_scores(), funcție, 400
ENUM, tip de date, 104, 121-125, 522
comparație cu SET, 124
indexare, 125
valori admise, 122
err(), metodă (Perl DBI), 672
error_reporting(), funcție, 391
errstr(), metodă (Perl DBI), 672
escape(), metodă (modulul CGI.pm), 357
ESCAPED BY, clauză (instrucțiunea LOAD DATA), 579-580
escapeHTML(), metodă (modulul CGI.pm), 357
etichetă de deschidere scurtă, stil (scripturi PHP), 375
etichete stil HTML (PHP), 680-681
evaluarea expresiilor, 125. *Vezi și* selectare
evitarea interacțiunii cu serverul MySQL, 492
executarea
fișiere de descriere, 47
interogări, privilegii de acces, 480-481
mysql, 92-93
mod grup, 95-96
scripturi de verificare a tabelelor, 496-497
--execute, opțiune de program, 612
execute(), metodă (Perl DBI), 670
execuție
copii de siguranță, copii directe, 460
operații de golire, 454
repararea tabelelor, 490-491
transferul în fișiere text, 457-458
verificări aprofundate ale tabelelor, 490
exemplul cu Liga istorică
automatizarea procesului de notificare privind plata cotizației, 337-343
browser pentru interese comune, 370-371
editarea intrărilor aferente membrilor, 344-349
editarea on-line a intrărilor, 405-412
generarea catalogului, 330-337
generarea documentului HTML, 350-352
identificarea intereselor comune, 349-350
testul cu președinții, 403-404
existența tabelelor, verificare, 188-189
--exit-info, opțiune de program, 623
exit, comandă (clientul mysql), 615
EXP(), funcție numerică, 540
expirarea fișierelor jurnal, 454
EXPLAIN, instrucțiune, 202, 573-574
explode(), funcție, 403
EXPORT_SET(), funcție șir, 544
expresii, 71
conversia de tip, 152-155
forțare, 156
testare, 155-156
valori incorecte, 157
evaluare, 144
NOW(), funcție, 41

operatori, 146
de comparaţie, 147-149, 150
logici, 146
pe biţi, 146
precedenţă, 150
referinţe la coloane, 146
rezultatele interogărilor, calcul, 71
scriere, 145
valori dată-oră (reguli de interpretare), 155
valori NULL, 151
expresii simple, 71
--extend-check, opţiune de program, 606
--extended-insert, opţiune de program, 629
extended-status, comandă (utilitarul mysqladmin), 619
extensii (tipuri de fişiere), 424
EXTRACT(), funcţie, 554

F

-f, opţiune de program, 606, 606, 610, 613, 168, 629, 631
--fast, opţiune de program, 608
fetch(), metodă (Perl DBI), 305, 670
fetchall_arrayref(), metodă (Perl DBI), 312, 670
fetchrow_array(), metodă (Perl DBI), 299, 305-306, 311, 671
fetchrow_arrayref(), metodă (Perl DBI), 305-306, 311-312, 671
fetchrow_hashref(), metodă (Perl DBI), 305-308, 671
FIELD(), funcţie şir, 545
FIELDS, clauză (instrucţiunea LOAD DATA), 579
--fields-enclosed-by, opţiune de program, 630, 632
--fields-escaped-by, opţiune de program, 630, 632
--fields-optionally-enclosed-by, opţiune de program, 630, 632
--fields-terminated-by, opţiune de program, 630, 632
FILE, privilegiu, 476, 575
find, comandă, 421
FIND_IN_SET(), funcţie şir, 545
finish(), metodă (Perl DBI), 299, 671
fire de execuţie
instrucţiunea KILL, 577-578
instrucţiunea SHOW PROCESSLIST, 590-591
fişier istoric (client mysql), 611
fişiere
acces la citire (securitate), 470
blocare, 492-494
copie directă, 456
crontab
editare, 454
script de verificare a tabelelor, 496-497
de pornire, amplasare, 440-441
executarea copiilor de siguranţă, 453
fişiere descriere, execuţie, 47
fişiere jurnal
expirare, 454
rotaţie, 453
interogări SQL, stocare, 42
protecţie, 470
refacere din copiile de siguranţă, 461-462
fişiere antet, mysql.h (API C), 634
fişiere copie directă, 456
fişiere cu opţiuni, 91-92, 601-603
citire pentru parametri de conexiune, 254-257
specificare în scripturi Perl, 319-322
fişiere cu opţiuni specifice utilizatorilor, 603
fişiere de date, 423
fişiere de grup (mysql), executare, 92-93
fişiere de stare, 428-431
modificarea amplasamentului, 435
fişiere descriere, execuţie, 47
fişiere formular, 424
fişiere include (scripturi PHP), 380-384
fişiere jurnal
expirare, 454
întreţinere, 452
protecţie, 470
rotaţie, 453
sisteme Linux, 454
fişierul my.cnf, creare, 91
FLOAT, tip de date, 103, 106-107, 518
FLOOR(), funcţie, 74, 540
--flush, opţiune de program, 625
--flush-logs, opţiune de program, 459, 629
FLUSH, instrucţiune, 451, 574
FLUSH PRIVILEGES, instrucţiune, 481
flush-hosts, comandă (utilitarul mysqladmin), 619
flush-logs, comandă (utilitarul mysqladmin), 619
flush-privileges, comandă (utilitarul mysqladmin), 619
flush-status, comandă (utilitarul mysqladmin), 619
flush-tables, comandă (utilitarul mysqladmin), 619
flush_time, variabilă (serverul MySQL), 464
FOR, clauză (instrucţiunea SET), 586
--force, opţiune de program, 606, 610, 613, 168, 629, 631
FORMAT(), funcţie şir, 545
formatare
coloane, 47
dată-oră, 127-132
date de ieşire, 219-220
text, RTF (Rich Text Format), 31
forţarea conversiei de tip, 156
fracţii (numere cu virgulă mobilă), 106
fragmentare
a tabelelor, 208
reducere, 120
FROM, clauză
instrucţiunea SELECT, 62, 584
instrucţiunea SHOW, 588
FROM_DAYS(), funcţie, 554
FROM_UNIXTIME(), funcţie, 554
FTP (protocol de transfer al fişierelor), selectare ISP, 706
--full, opţiune de program, 629
func(), metodă (Perl DBI), 672
funcţii, 509-510, 521. *Vezi şi* metode
add_new_event(), 398
addslashes(), 392
API C, 640
administrative, 658-659
construcţie a interogărilor, 646-647
de gestiune a conexiunii, 640-645
de raportare a erorilor, 645
depanare, 660-662
informaţii, 657-658
prelucrare a setului de rezultate, 648-657
argumente, separare, 145
AVG(), 80-81
check_response(), 404
CONCAT(), 152
COUNT(), 77-79
CREATE FUNCTION, instrucţiune, 567
CRYPT(), 479
dată şi oră, 550-557
DATE_ADD(), 74
DATE_SUB(), 74
DAYOFMONTH(), 72-73
de comparaţie, 537
die(), 384-385
display_cell(), 397

display_entry(), 410-411
display_scores(), 398
diverse, 559-563
DROP FUNCTION, instrucţiune, 572
enter_scores(), 400
error_reporting(), 391
explode(), 403
expresii
conversie de tip, 152-155
operatori, 145-149, 150
scriere, 145
valori dată-oră, 155
valori NULL, 151
fişiere include, 380-384
FLOOR(), 74
get_tty_password(), 265
getenv(), 257
getopt_long(), 258, 260-261
hidden_field(), 404
html_begin(), 383
html_end(), 383
implode(), 403
invocare (sintaxă), 145
LAST_INSERT_ID(), 116, 187-188
load_defaults(), 254-257
MAX(), 80-81
MIN(), 80-81
MONTH(), 72-73
mysql_fetch_array(), 390
mysql_fetch_field(), 411-412
mysql_fetch_object(), 390
mysql_pconnect(), 376-378
mysql_query(), 378, 386-387
NOW(), 41
nullable(), 412
numerice, 539-542
paranteze, 42
PASSWORD(), 479, 484
Perl DBI, 674
PHP, 680-681
construcţia interogărilor, 685-687
de baze de date, 700-701
gestiunea conexiunii, 682-683
preluarea rândurilor, 389-390
prelucrarea seturilor de rezultate, 687-670
raportarea erorilor, 683-684
PROCEDURE_ANALYSE(), 135, 207
process_result_set(), 270
require(), 382
solicit_event_info(), 399
string(), 543-550
SUM(), 80-81
sumar (agregat), 558-559
TO_DAYS(), 73
YEAR(), 72-73
funcţii administrative (API C), 658-659
funcţii agregat, 558-559
funcţii de comparaţie, 537
funcţii de construcţie a interogărilor
API C, 646-647
API PHP, 685-687
funcţii de depanare (API C), 659-660
funcţii de gestiune a conexiunilor
API C, 640-645
API PHP, 681-682
funcţii de informaţie (API C), 657-658
funcţii de prelucrare a setului de rezultate
API C, 648-657
API PHP, 687-670
funcţii de raportare a erorilor
API C, 645
API PHP, 683-684
funcţii de stare
API C, 645
API PHP, 683-684
funcţii de sumar, 558-559
funcţii pentru baze de date (API PHP), 700-701
funcţii scoase din uz (API C), 660-662
funcţii şir, 543-550
furnizori de servicii Internet. *Vezi* ISP
fus orar, configurare, 442

G

\g, comandă (clientul mysql), 615
găsire
amplasarea catalogului de date, 420-421
cataloagele fişierelor de pornire, 440-441
exemple de programe client, 238
fişiere nesigure, 470-471
interese comune (exemplul cu Liga istorică), 349-350
înregistrări, 26-27
generare
etichete HTML, 383
numere unice, 58, 110
AUTO_INCREMENT, 112-113
pagini Web din scripturi PHP, 373-374
secvenţe, 116
gestiunea conturilor de utilizator, 444-445
GET_LOCK(), funcţie, 561
get_tty_password(), funcţie, 265
getenv(), funcţie, 257
getopt_long(), funcţie, 257, 260-261
ghicitoarea cu preşedinţi americani (Liga istorică americană), 401-402, 403-404
ghilimele duble, 101
go, comandă (clientul mysql), 615
golire întârziată a indexului, 211
GRANT, instrucţiune, 483-484, 574-576
efectul asupra tabelelor de acordare, 444-445
sintaxă, 445-457
WITH GRANT OPTION, clauză, 451
GRANT, privilegiu, 476
GREATEST(), funcţie de comparaţie, 537
GROUP BY, clauză (instrucţiunea SELECT), 79, 585
gruparea operatorilor, 527

H

\h, comandă (clientul mysql), 615
-h, opţiune de program, 599, 613, 617, 622
HAVING, clauză (instrucţiunea SELECT), 80, 585
HEAP, format de stocare a tabelelor, 166
--help, opţiune de program, 598, 599
help, comandă (clientul mysql), 615
HEX(), funcţie şir, 545
HIGH_PRIORITY, cuvânt cheie, 213, 583
--host, opţiune de program, 599
Host, coloană (coloane de definire a domeniului de valabilitate), 478
HOSTS, cuvânt cheie, 574
HOUR(), funcţie dată şi oră, 555
--howto, opţiune de program, 617
html_begin(), funcţie, 383
html_end(), funcţie, 383

I-Î

-i, opţiune de program 597, 613, 168, 631, 633
identificarea echivalenţelor cu un model, 75-76, 149
operatori, 534-537
simbolul procentului (%), caracter de înlocuire, 447, 534
identificarea valorilor de date, 45
identificator de proces (PID), fişier, 428
IDENTIFIED BY, clauză (instrucţiunea GRANT), 575

IF EXISTS, clauză
instrucțiunea DROP DATABASE, 572
instrucțiunea DROP TABLE, 169
IF NOT EXISTS, clauză (instrucțiunea CREATE TABLE), 166, 567
IF(), funcție de comparație, 537-538
IFNULL(), funcție de comparație, 538
--ignore, opțiune de program, 631
--ignore-space, opțiune de program, 613
IGNORE, cuvânt cheie, 571, 577
IGNORE LINES, clauză (instrucțiune LOAD DATA), 580
implode(), funcție, 403
import
date în tabele, 61-62
înregistrări, 96
IN, operator de comparație, 531
include(), funcție, 382
INDEX, privilegiu, 476, 575
indexarea coloanelor
coloane BLOB, 120
coloane TEXT, 120
efect asupra selecției tipului, 143
indexuri, 169, 195-196
avantaje, 196-199
caracteristici, 169-170
convenții de nume, 161-162
creare, 170-172
CREATE INDEX, instrucțiune, 567
dezavantaje, 199
DROP INDEX, instrucțiune, 573
golire întârziată, 211
încărcarea datelor, optimizare, 210
îndrumări pentru utilitarul de optimizare a interogărilor, 202-205
selectarea coloanelor pentru, 199-201
SHOW INDEX, instrucțiune, 589-590
sintetice, 208
ștergere, 172-173
indexuri hash, 208
indexuri sintetice, 208
indicatori spre variabile de tratare a conexiunilor (neinițializați), depanare, 290-291
influențe lingvistice (PHP), 380
info, atribut (Perl DBI), 675
--information, opțiune de program, 606
--init-file, opțiune de program, 623
inițializare
baza de date demonstrativă, 43-44
baza de date mysql (sisteme UNIX), 507
catalog de date, 437
INNER JOIN, cuvânt cheie, 584
INSERT, instrucțiune, 60-61, 99, 576-577
privilegii de super-utilizator, acordare, 484-485
INSERT, privilegiu, 476, 575
INSERT(), funcție șir, 545
INSERT ID, cuvânt cheie, 586
inserție
înregistrări, 60
rânduri în tabele, 60-61
instalare
distribuția samp_db, 515
MySQL, 38, 500-501
pachete, 501
servere multiple, 466
sisteme UNIX, 503
Apache, 510-512
conturi de utilizator, creare, 504
distribuție, instalare, 504-507
distribuții, obținere, 504-505
inițializarea bazei de date mysql, 507
PHP, 510-512
suportul Perl DBI, 509-510
sistemele de operare Windows, 512
Apache, 515
distribuția client-server, 513-514
MyODBC, 514
PHP, 515
suportul Perl DBI, 514-515
versiune, selectare, 502
--install, opțiune de program, 625
install_driver(), metodă (Perl DBI), 667
INSTR(), funcție șir, 545
instrucțiuni (SQL)
acceptate/neacceptate în MySQL, 158-160
ALTER TABLE, 90, 173-175, 424, 564-566
crearea indexurilor, 170-172
ștergerea indexurilor, 172-173
comentarii în, 181
CREATE, privilegiu, 476, 575
CREATE DATABASE, instrucțiune, 43, 163, 423, 566
CREATE FUNCTION, instrucțiune, 567
CREATE INDEX, instrucțiune, 170-172, 424, 567
CREATE TABLE, instrucțiune, 47, 57, 105, 164-165, 424, 567-571
crearea condiționată a tabelelor, 166
rezultatele interogării SELECT, crearea tabelelor pornind de la, 167-169
specificatori ai tipului de stocare, 165-166
tabele temporare, 167
DELETE, 88, 451, 571-572
atributul AUTO_INCREMENT, 114
DESCRIBE, 49-50, 176, 289, 572
trunchierea valorilor, 109
DROP DATABASE, 163, 423, 572
DROP FUNCTION, instrucțiune, 572
DROP INDEX, instrucțiune, 172-173, 424, 573
DROP TABLE, instrucțiune, 169, 424, 573
EXPLAIN, 202, 573-574
fișiere descriere, execuție, 47
FLUSH, 451, 574
FLUSH PRIVILEGES, 481
GRANT, instrucțiune, 483-484, 574-576
efectul asupra tabelelor de acordare, 444-445
sintaxă, 445-457
WITH GRANT OPTION, clauză, 451
INSERT, 60-61, 99, 576-577
KILL, 577-578
LOAD DATA, 62, 96, 209, 453, 578-581
LOCK TABLE, 493
LOCK TABLES, 190-191, 581
OPTIMIZE TABLE, 208, 582
prompturi, 43
REPLACE, 582
REVOKE, 582-583
efectul asupra tabelelor de acordare, 444-445
sintaxă, 451
SELECT, 62, 177, 583-586
clauze, 63
crearea tabelelor din rezultate, 167-169
GROUP BY, clauză, 79
HAVING, clauză, 80
LIMIT, clauză, 70
ORDER BY, clauză, 69
uniri complete, 178-179
uniri la stânga, 180-181
uniri normale, 178
WHERE, clauză, 64-65

sensibilitatea la diferența între majuscule și minuscule, 162
SET, 586-588
SHOW, 588
SHOW COLUMNS, 589
SHOW COLUMNS FROM, 176
SHOW DATABASES, 50, 176, 424, 589
SHOW FIELDS FROM, 289
SHOW GRANTS, 589
SHOW INDEX, 589-590
SHOW INDEX FROM, 176
SHOW PROCESSLIST, 590-591
SHOW STATUS, 591-593
SHOW TABLE STATUS, 176, 593-594
SHOW TABLE STATUS FROM, 176
SHOW TABLES, 50, 176, 425, 594
SHOW TABLES FROM, 176
SHOW VARIABLES, 594
stocare, 42
terminare, 41
tranzacții, 189-192
UNLOCK TABLES, 190-191, 594
UPDATE, 90, 595
USE, 43, 163, 595
instrucțiuni de afișare, depanarea scripturilor Perl, 322-323
instrucțiuni SELECT imbricate. *Vezi* subselecții
instrumente. *Vezi* instrumente create de terțe părți
instrumente administrative create de terți, 703
instrumente create de terțe părți, 702
asistență ODBC, 702
instalare, 500-501
instrumente administrative, 703
instrumente de conversie a datelor, 702
interfețe API de limbaj, 702
interfețe de program, 703
instrumente de conversie a datelor create de terți, 702
instrumente de programare (create de terți), 702
INT, tip de date, 103, 106-107, 518
integrarea sistemului MySQL
cu aplicațiile, 220-221
cu aplicațiile Web, 220-222
integritate referențială, 192-193
intercorelarea tipurilor de coloane, 143
interese comune (exemplul cu Liga istorică), găsire, 349-350
interfața API mSQL, conversie la interfața API MySQL, 224
interfața Perl DBI, 222-226, 663
atribute, 675
atribute dinamice, 675
atribute generale ale variabilelor de manipulare, 675
atributele variabilelor de manipulare a bazelor de date, 675
atributele variabilelor de manipulare a instrucțiunilor, 676-679
cerințe de sistem, 293-294
exemplul cu Liga istorică
automatizarea procesului de înștiințare privind plata cotizațiilor, 337-343
editarea intrărilor aferente membrilor, 344-349
generarea ca document HTML, 350-352
generarea catalogului, 330-337
identificarea intereselor comune, 349-350
factori de selectare
interpretori autonomi sau modulari, 231-233
limbaje compilate sau interpretate, 230
mediu de execuție, 229
portabilitate, 235-236
timp de dezvoltare a programului, 233-235
funcții utilitare, 674
instalare (sisteme UNIX), 509-510
metode, 665-667
func(), 672
metode de clasă, 665-667
metode de manipulare a bazelor de date, 667-669
metode de manipulare a instrucțiunilor, 669-672
metode generale de manipulare, 672-673
scrierea scripturilor, 664
scripturi, 294
asocierea parametrilor, 318-319
câmpuri de înlocuire, 316-319
cicluri de preluare a rândurilor, 305-308
citarea șirurilor, 314-316
comparație cu PHP, 379
depanare, 322
depanare cu parametri de afișare, 322-323
depanare cu urmărire, 324-325
exemplu, 295-300
metadatele setului de rezultate, 325-330
număr de rânduri returnate, 308
parametri de conexiune, 319-322
prelucrarea înregistrărilor fără seturi de rezultate, 304-305
rezultate cu un singur rând, 380-310
seturi de rezultate complete, 310-313
tratarea erorilor, 300-303
valori NULL, 313-314
scripturi bazate pe Web, 353-354
browser demonstrativ pentru baze de date, 362-366
browser pentru interese comune (exemplul cu Liga istorică), 370-371
browser pentru punctaje din proiectul pentru evidența rezultatelor școlare, 366-369
conectarea la serverul MySQL, 360-361
configurarea serverului Apache, 354-355
scripturi CGI.pm, 355-360
tipuri de date, 294-295
variabile de mediu, 679
interfețe (create de terți), 703
interfețe API pentru diverse limbaje (instrumente create de terțe părți), 702
interfețe de program (instrumente create de terțe părți), 703
Internet (cerere de căutare bazată pe Web), 28
interogări
abreviere, 93-94
copiere și lipire, 95
CREATE TABLE, instrucțiune, 106
date, regăsire, 72-74
DESCRIBE, instrucțiune (trunchierea valorilor), 108
determinare, 83
emitere, 40
fișiere jurnal, protecție, 470
formular complet determinat, 83
funcții (paranteze), 42
mărimea literei, 42
operatori AND, 66
pe mai multe linii, 42
PHP
citare, 392
emitere, 386-387
eșec, motive de, 386-387

prelucrarea rezultatelor, 387-390
prompturi, 43
rezultate
calcul, 71
eliberarea memoriei alocate, 378
identificarea echivalenţelor cu un model, 75-76
limitare, 70
restricţionare, 80
rezumare, 76-82
sortare, 68-69
SELECT, instrucţiune, 62, 64
stocare, 42
tabele multiple, specificare, 82-88
verificarea accesului, 480-481
interogări, prelucrare. *Vezi* prelucrarea interogărilor
interogări complet determinate, 83
interogări cu mai multe linii, anulare, 42
interogări reuşite (PHP), 386-387
interpretare
valori ambigue ale anilor, 132
valori dată-oră (reguli), 155
interpretoare (shell-uri)
aliasuri, 92-93
istoric al comenzilor, 92
scripturi, scriere, 96
interpretoare autonome, comparaţie cu interpretoare modulare, 231-233
interpretoare modulare, comparaţie cu interpretoare autonome, 231-233
INTERVAL(), funcţie de comparaţie, 538
INTO, cuvânt cheie, 577
INTO OUTFILE, clauză (instrucţiune SELECT), 584
intrări aferente membrilor (exemplul cu Liga istorică), editare, 344-349
introducere
funcţii (paranteze), 41-42
instrucţiunea CREATE TABLE, 47
interogări, 41
mărimea literelor, 42
pe mai multe linii, 41
scurtături, 93-94
opţiuni în linia de comandă, 39-40
parole, 39-40
punctajele elevilor (proiectul de evidenţă a rezultatelor şcolare), 393-401
introducere de la tastatură
funcţii (paranteze), 42
instrucţiunea CREATE TABLE, 47
interogări, 41
cu mai multe linii, 42
mărimea literelor, 42
scurtături, 93-94
parole, 39-40
inventar, cereri de căutare bazate pe Web, 28
invocare
comanda mysqld, 440
comenzi, 438
funcţii (sintaxă), 145
isamchk (sintaxă), 489-490
scriptul mysql.server, 440-441
scriptul safe_mysqld, 440-441
utilitarul mysqlimport, 62
IS NOT NULL, operator de comparaţie, 531
IS NULL, operator de comparaţie, 531
IS_BLOB(), macrocomandă accesoriu (API C), 640
IS_NOT_NULL(), macrocomandă accesoriu (API C), 639
IS_NUM(), macrocomandă accesoriu (API C), 639
IS_PRI_KEY(), macrocomandă accesoriu (API C), 639
ISAM, tabele
copiere, 456
format de stocare, 166
isamchk, utilitar, 415, 597, 604
interacţiunea cu serverul, 492
opţiuni specifice acceptate, 605-607
opţiuni standard acceptate, 605
repararea tabelelor, 491
sintaxă de invocare, 489-490
utilizare, 605
variabile, 608-609
verificarea tabelelor, 490
ISNULL(), funcţie de comparaţie, 538
ISP (furnizori de servicii Internet)
MySQL, cerinţe de instalare, 38
selectare, 704-705
consideraţii privind lărgimea de bandă, 706
criterii generale, 707-711
criterii specifice sistemului MySQL, 712-714
determinarea necesităţilor dumneavoastră, 705-706
servicii furnizate, 706
înălţime, afişare, 136-137
încărcare
date, optimizare, 209-211
înregistrări în tabele, 62
închidere automată, server MySQL, 442-443
închidere manuală (server MySQL), 442-443
închiderea conexiunilor cu serverul, 41
închiderea serverului MySQL, 438, 442-443
înregistrarea numelui de domeniu, selectare ISP, 706
înregistrări
acces de la distanţă, 28
actualizare, 88-90
adăugare, 60
duplicare, eliminare, 76-77
găsire, 27
importare, 96
încărcare în tabele, 62
înlocuire, 59
numere unice, atribuire, 53
regăsire, 62
din tabele multiple, 82-88
identificarea echivalenţelor cu un model, 75-76
specificarea criteriilor, 64-67
valoarea NULL, 67-68
rezumare, 27, 76-82
sortare, 27, 68-69
ştergere, 88
valori NULL, sortare, 69
înregistrări duplicate, eliminare, 76-77
înscrierea pe lista de corespondenţă, 502
întreţinere
atribuţii administrative, 418
planificare, 496
script de verificare a tabelelor, 496-497
întreţinere preventivă, 418
planificare, 496
întreţinerea fişierelor jurnal, 452

J-K

-j, opţiune de program, 610
--join, opţiune de program, 610
JOIN, cuvânt cheie, 584
jurnal de eroare, 428
jurnal general, 428-429, 452
activare, 453
rotaţie (rularea scriptului), 454
jurnale de actualizare, 428-429
activare, 453
rotaţie, 455
scripturi de expirare, 455
-k, opţiune de program, 606, 633
KEY, clauză (instrucţiunea CREATE TABLE), 569
key_buffer_size, parametru server, 214
key_buffer_size, variabilă (serverul MySQL), 464

--keys, opţiune de program, 633
--keys-used, opţiune de program, 606
kill, comandă, 444
utilitarul mysqladmin, 619-620
KILL, instrucţiune, 577-578

L

-l, opţiune de program, 606, 613, 623, 629, 632
--language, opţiune de program, 623
LAST_INSERT_ID, cuvânt cheie, 586
LAST_INSERT_ID(), funcţie, 187-188, 562
secvenţă, generare, 116
lărgime de bandă, selectare ISP, 706, 709
lăţime (coloane), declarare, 107
lăţime maximă (coloane), declarare, 107
LCASE, funcţie şir, 546
LEAST(), funcţie de comparaţie, 538
LEFT(), funcţie şir, 546
LEFT JOIN, cuvânt cheie, 85, 584
LEFT OUTER JOIN, cuvânt cheie, 181
legare
biblioteca client API C, 634
programe client, 239-240
tabele, 53
LENGTH(), funcţie şir, 546
Liga istorică americană
editarea on-line a intrărilor, 405-412
scenariu exemplu, 29-30
tabele, 44
creare, 47-49
tabelul cu membri, 46-47
tabelul cu preşedinţi, 45-46
LIKE, operator, 75-76, 534-535
limbaje compilate, comparaţie cu limbaje interpretate, 230
limbaje de scripting, 31
limbaje interpretate
comparaţie cu limbajele compilate, 230
interpretoare autonome comparate cu interpretoarele modulare, 231-233
limbajele interogărilor, 35
LIMIT, clauză
instrucţiunea DELETE, 571-572
instrucţiunea SELECT, 70, 584
instrucţiunea UPDATE, 595
limitare
accesul utilizatorilor, 447
rezultatele interogării, 69
HAVING, clauză, 80
--lines-terminated-by, opţiune de program, 630, 632
LINES, clauză (instrucţiunea LOAD DATA), 579
LINES TERMINATED BY, clauză (instrucţiunea LOAD DATA), 580
linie de comandă
criterii de regăsire, specificare, 64-67
funcţii (paranteze), 41-42
instrucţiuni, terminare, 41-42
interogări
emitere, 40
mărimea literelor, 41-42
pe mai multe linii, 41-42
rezultate, stocare, 68-69
stocare, 42
verificarea accesului, 480-481
interogări cu mai multe linii, anulare, 41-42
opţiuni, 39-40
opţiuni. *Vezi şi* programe; opţiuni
analiză, 257-266
MySQL, 253-254
parole, introducere, 39-40
prompturi, 43
QUIT, comandă, 41
liniuţă de subliniere, caracter de înlocuire pentru identificarea echivalenţei cu un model, 534
Linux (rotaţia fişierelor jurnal), 454
lipirea interogărilor, 95
liste de corespondenţă, înscriere, 502
liste de discuţii, înscriere la, 502
LOAD DATA, instrucţiune, 62, 96, 209, 453, 578-581
load_defaults(), funcţie, 254-257
LOAD_FILE(), funcţie, 562
--local, opţiune de program, 632
LOCAL, cuvânt cheie, 62, 578
localizare
cataloagele cu fişierele de pornire, 440-441
fişiere nesigure, 470-471
înregistrări, 26-27
LOCATE(), funcţie şir, 546
--lock-tables, opţiune de program 620, 632
LOCK TABLES, instrucţiune, 190-191, 493, 581
--log, opţiune de program, 623
--log-isam, opţiune de program, 623
--log-long-format, opţiune de program, 623
--log-update, opţiune de program, 623
LOG(), funcţie numerică, 540
LOG10(), funcţie numerică, 541
LOGS, cuvânt cheie, 574
LONGBLOB, tip de date, 84, 505. *Vezi şi* BLOB, tip de date
LONGTEXT, tip de date, 84, 506. *Vezi şi* TEXT, tip de date
looks_like_number(), funcţie (Perl DBI), 674
--low-priority-updates, opţiune de program, 623
LOW_PRIORITY, cuvânt cheie, 212, 572, 576-577, 595
LOWER(), funcţie şir, 546
LPAD(), funcţie şir, 546
ls. comandă, 462
LTRIM(), funcţie şir, 546
lungimea rândurilor, selectare, 118

M

-m, opţiune de program, 608
macrocomenzi (API C), 639
macrocomenzi accesoriu (API C), 639
mai mare sau egal, semn (>=), operator de comparaţie, 530
mai mic (<), operator de comparaţie, 530
mai mic sau egal (<=), operator de comparaţie, 530
MAKE_SET(), funcţie şir, 547
manipularea datelor de intrare, personalizare, 218-219
manual, 37
baza de date demonstrativă
creare, 43-44
obţinere, 38
Liga istorică americană
editarea on-line a intrărilor, 405-412
ghicitoarea cu preşedinţii, 401-402, 403-404
proiect de evidenţă a rezultatelor şcolare
introducerea punctajelor elevilor, 393-401
tabelul cu absenţe, 59
tabelul cu elevi, 55-56
tabelul cu evenimente, 52-55, 58
tabelul cu punctaje, 50-51, 59
MAX(), funcţie, 80-81, 559
max_allowed_packet, variabilă (server MySQL), 464
max_connections, variabilă (server MySQL), 464
MAX_ROWS, cuvânt cheie, 570
mărimea literelor (interogări), 41-42

MD5(), funcție, 562
mediu de execuție (factori în selectarea API), 229
--medium-check, opțiune de program, 608
MEDIUMBLOB, tip de date, 84, 504. *Vezi și* BLOB, tip de date
MEDIUMINT, tip de date, 103, 106-107, 518
cerințe privind spațiul de stocare, 107
MEDIUMTEXT, tip de date, 84, 505. *Vezi și* TEXT, tip de date
memorarea în cache a interogărilor (motoare de baze de date), 318
mesaje de eroare, eliminarea în scripturile PHP, 377-378
metadate (set de rezultate), prelucrare
interogări, 278-283
în scripturi Perl, 325-330
metoda legăturii simbolice (modificarea amplasamentului bazelor de date), 434
metode. *Vezi și* funcții
modulul CGI.pm
escape(), 357
escapeHTML(), 357
param(), 356
Perl DBI, 294-295, 665-667
func(), 672
metode de clasă, 665-667
metode de manipulare a bazelor de date, 667-669
metode de manipulare a instrucțiunilor, 669-672
metode generale de manipulare, 672-673
metode de clasă, (Perl DBI), 665-667
metode de manipulare a instrucțiunilor (Perl DBI), 669-672
metode generale de manipulare (Perl DBI), 672-673
MID(), funcție șir, 547
MIN(), funcție, 80-81, 559
MIN_ROWS, cuvânt cheie, 570
MINUTE(), funcție, 555
mod grup (mysql), executare, 95-96, 611
MOD(), funcție numerică, 541
mod_perl, modul, 232
mod_php3, modul, 232
modificare
script de pornire, 442
structura tabelelor, 91
modificarea amplasamentului catalogului de date, 431-435
baze de date, 434
estimarea efectului, 432-433
fișiere de stare, 435
tabele, 434
modificarea semnificației caracterelor de citare, 101
MODIFY, clauză (instrucțiunea ALTER TABLE), 174
modul interactiv
client mysql, 611
mysql, 95-96
modulo (%), operator, 528
MONTH(), funcție, 72-73, 555
MONTHNAME(), funcție, 555
my_bool, tip de date scalare, (API C), 635
my_ulonglong, tip de date scalare (API C), 635
MyISAM, format de stocare al tabelelor, 166
myisamchk, utilitar, 415, 597, 604
interacțiunea cu serverul, 492
opțiuni specifice acceptate, 605-607
opțiuni standard acceptate, 605
repararea tabelelor, 491
sintaxă de invocare, 489-490
utilizare, 604-605
variabile, 608-609
verificarea tabelelor, 490
versiuni viitoare, 415, 597, 604-605
myisampack, utilitar, 597, 609
opțiuni specifice acceptate, 610
opțiuni standard acceptate, 609
utilizare, 609
MyODBC, 702
instalare, 514
MySQL
administrare
atribuții generale de administrare, 416-417
depanare, 418
întreținere, 418
securitate, 417
trecere în revistă, 416
avantaje, 13-14
caracteristici neacceptate, 189-194
caracteristici SQL acceptate/neacceptate, 158-160
conversie la, 14-15
instalare, 38, 500-501
instrumente incluse în distribuție, 15
integrare cu aplicații, 220-221
integrare cu aplicațiile Web, 220-222
interfețe API, 222-224
C, 222-224
factori pentru selectare, 228-236
Perl DBI, 222-226
PHP, 222-223, 227-228
probleme legate de licență, 15-16, 31
sesiuni, terminare, 41
situri oglindă, 501
utilitare, 218-219
MySQL, catalog de date. *Vezi* catalog de date
MYSQL, structură (API C), 635
MySQL, utilitar de optimizare, 201-202
redefinire, 205-206
trecere în revistă a operațiilor, 202-205
mysql (program client), 597, 611
biblioteca GNU Readline, 93-94
comenzi, 615
comparație cu MySQL, 38
executare, 92-93
interogări, copiere și lipire, 95
limitări în sintaxa comentariilor, 595-596
manual, 37
mod grup, 95-96
opțiuni specifice acceptate, 612, 614
opțiuni standard acceptate, 612
tabele de acordare, 470
utilizare, 612
variabile, 614
mysql, utilitar, 218, 415
comparație cu programarea pentru MySQL, 219-220
mysql. bază de date (sisteme UNIX), inițializare, 507
mysql.h, fișier antet (API C), 634
mysql.server, script, 597, 616
invocare, 440-441
mysql_affected_rows(), funcție
API C, 267, 648
API PHP, 687-688
mysql_change_user(), funcție (API C), 640
mysql_close(), funcție
API C, 242, 640-641
API PHP, 681-682
mysql_connect(), funcție
API C, 661
API PHP, 682-683
mysql_create_db(), funcție
API C, 661
API PHP, 700-701
mysql_data_seek(), funcție
API C, 661
API PHP, 700-701
mysql_db_query(), funcție (API PHP), 685
mysql_debug, funcție (API C), 659

MYSQL_DEBUG, variabilă de mediu, 603
mysql_drop_db(), funcție
API C, 661
API PHP, 701
mysql_dump_debug_info(), funcție (API C), 660
mysql_eof(), funcție (API C), 661
mysql_errno(), funcție
API C, 244, 645
API PHP, 683-684
mysql_error(), funcție
API C, 244, 645
API PHP, 683-684
mysql_escape_string(), funcție (API C), 287-288, 646
mysql_fetch_array(), funcție, 389-390
mysql_fetch_field(), funcție, 411-412
API C, 278, 649
API PHP, 690-691
mysql_fetch_field_direct(), funcție (API C), 650
mysql_fetch_fields(), funcție (API C), 649-650
mysql_fetch_lengths(), funcție, 411-412
API C, 278, 650
API PHP, 691-692
mysql_fetch_object(), funcție, 389
API PHP, 692
mysql_fetch_object(), funcție, 390
mysql_fetch_row(), funcție, 389
API C, 269-270, 651
API PHP, 692
MYSQL_FIELD, structură (API C), 636-639
macrocomenzi accesoriu, 639
mysql_field_count(), funcție (API C), 272-273, 652-653
mysql_field_flags(), funcție (API PHP), 696-697
mysql_field_len(), funcție (API PHP), 697
mysql_field_name(), funcție (API PHP), 693
MYSQL_FIELD_OFFSET, tip de date scalar (API C), 635
mysql_field_seek(), funcție
API C, 653
API PHP, 693-694
mysql_field_table(), funcție (API PHP), 695
mysql_field_tell(), funcție (API C), 653
mysql_field_type(), funcție (API PHP), 695-696
mysql_free(), funcție (API C), 662
mysql_free_result(), funcție
API C, 269-270, 654
API PHP, 696-697
mysql_get_client_info(), funcție (API C), 657
mysql_get_host_info(), funcție (API C), 657
mysql_get_proto_info(), funcție (API C), 657
mysql_get_server_info(), funcție (API C), 657
MYSQL_HISTFILE, variabilă de mediu, 604
MYSQL_HOST, variabilă de mediu, 604
mysql_info(), funcție (API C), 654
mysql_init(), funcție (API C), 241, 641
MYSQL_INIT_COMMAND, opțiune de conexiune, 641
mysql_insert_id(), funcție
API C, 654
API PHP, 697-698
mysql_insertid, atribut (Perl DBI), 675
mysql_is_blob, atribut (Perl DBI), 678
mysql_is_key, atribut (Perl DBI), 678
mysql_is_not_null, atribut (Perl DBI), 678
mysql_is_num, atribut (Perl DBI), 678
mysql_is_pri_key, atribut (Perl DBI), 678
mysql_kill(), funcție (API C), 658
mysql_list_dbs(), funcție
API C, 661
API PHP, 685-686
mysql_list_tables(), funcție
API C, 662
API PHP, 686
mysql_max_length, atribut (Perl DBI), 678
mysql_num_fields(), funcție
API C, 270, 272, 278, 654-655
API PHP, 698-699
mysql_num_rows(), funcție
API C, 272, 278, 654-655
API PHP, 698-699
MYSQL_OPT_COMPRESS, opțiune de conexiune, 642
MYSQL_OPT_CONNECT_TIMEOUT, opțiune de conexiune, 642
MYSQL_OPT_NAMED_PIPE, opțiune de conexiune, 642
mysql_options(), funcție (API C), 642-643
mysql_pconnect(), funcție (API PHP), 683
mysql_ping(), funcție (API C), 643
MYSQL_PWD, variabilă de mediu, 603
mysql_query(), funcție
API C, 266, 646-647
API PHP, 686-687
MYSQL_READ_DEFAULT_FILE, opțiune de conexiune, 642
MYSQL_READ_DEFAULT_GROUP, opțiune de conexiune, 642
mysql_real_connect(), funcție (API C), 241-242, 643-645
mysql_real_query(), funcție (API C), 266-267, 647
mysql_refresh(), funcție (API C), 659
mysql_reload(), funcție (API C), 662
MYSQL_RES, structură (API C), 635
mysql_result(), funcție (API PHP), 699-700
MYSQL_ROW, tablou (API C), 635
mysql_row_seek(), funcție (API C), 656
mysql_row_tell(), funcție (API C), 656
mysql_select_db(), funcție
API C, 647
API PHP, 701
mysql_shutdown(), funcție (API C), 659
mysql_stat(), funcție (API C), 658
mysql_store_result, atribut (Perl DBI), 678
mysql_store_result(), funcție (API C), 269, 656
comparație cu mysql_use_result(), 276-278, 648
mysql_table, atribut (Perl DBI), 679
mysql_tablename(), funcție (PHP API), 700
MYSQL_TCP_PORT, variabilă de mediu, 603
mysql_thread_id(), funcție (API C), 658
mysql_type_name, atribut (Perl DBI), 679
MYSQL_UNIX_PORT, variabilă de mediu, 604
mysql_use_result, atribut (Perl DBI), 679
mysql_use_result(), funcție (API C), 269, 657
comparație cu mysql_store_result(), 276-278, 648

mysqlaccess, script, 597, 616
opțiuni acceptate specifice, 616-617
opțiuni acceptate standard, 616
utilizare, 616
mysqladmin, utilitar, 218, 415, 597, 168
comenzi, 618-620
opțiuni acceptate specifice, 618
opțiuni acceptate standard, 168
utilizare, 168
utilizator rădăcină (configurarea parolelor), 437
mysqladmin variables, comandă, 420-421
mysqlbug, script, 597, 620-621
mysqld, comandă, invocare, 440
mysqld, server, 415, 597, 622
accesul la catalogul de date, 422-423
atribuții generale de administrare, 416-417
opțiuni acceptate specifice, 622-625
opțiuni acceptate standard, 621
utilizare, 621
variabile, 625-628
mysqldump, utilitar, 176, 218, 415, 456, 597, 628
opțiuni, 459-460
opțiuni acceptate privind formatul datelor, 630
opțiuni specifice acceptate, 628-630
opțiuni standard acceptate, 628
transferul în fișiere text, 457-458
utilizare, 628
variabile, 630
mysqlimport, utilitar, 62, 218, 597, 631
opțiuni acceptate privind formatul datelor, 632
opțiuni specifice acceptate, 631-632
opțiuni standard acceptate, 631
utilizare, 631
Mysqlperl, modul, 224
mysqlshow, utilitar, 176, 597, 633
opțiuni specifice acceptate, 633
opțiuni standard acceptate, 633
utilizare, 633

N

-n, opțiune de program 604, 623
NAME, atribut (Perl DBI), 676
NAME_lc, atribut (Perl DBI), 676
NAME_uc, atribut (Perl DBI), 676
NATURAL LEFT JOIN, cuvânt cheie, 181, 585
neat(), funcție (Perl DBI), 674
neat_list(), funcție (Perl DBI), 674
--new, opțiune de program, 623
--no-auto-rehash, opțiune de program, 613
--no-create-info, opțiune de program, 629
--no-data, opțiune de program, 629
--no-defaults, opțiune de program, 599
--no-symlinks, opțiune de program 597
NOT, operator logic, 533
NOT IN, operator de comparație, 531
NOT LIKE, operator de comparație, 534-535
NOT NULL, atribut, 111, 569
coloane șir, 126
NOT REGEXP, operator de identificare a echivalențelor cu un model, 535
NOT RLIKE, operator de identificare a echivalențelor cu un model, 536
notație științifică, 101
notație științifică corectă, 101
notație zecimală (numere cu virgulă mobilă), 100
NOW(), funcție dată-oră, 41, 555
NUL, octeți, 101
NULL, atribut
coloane numerice, 111
coloane șir, 126
NULL, cuvânt cheie, 569
NULL, valori, 48, 67-68, 102
aplicare în coloane, 90
în expresii, 151
prelucrarea cu scripturile Perl, 313-314
rezultatele interogărilor PHP, testare, 390
selectarea tipurilor de coloane, 141
sortare, 69
NULL, valori în coloane, depanare, 291-292
nullable(), funcție, 412
NULLABLE, atribut (Perl DBI), 676
NUM_OF_FIELDS, atribut (Perl DBI), 676
NUM_OF_PARAMS, atribut (Perl DBI), 676
nume complet determinate, 161
rezolvare, 448-449
nume de utilizator, 39
nume de utilizator prestabilit (Windows), 39
nume determinate, 161
numele gazdei locale, specificarea în tabelul de acordare, 448-449
numere cu virgulă mobilă, 100, 107
coloane, 106
trunchierea valorilor, 109
numere întregi (coloane), 106
numere unice
atribuire, 53, 112
generare, 59, 112-113
secvențe
generare, 116
valori negative, 114
numerice, funcții, 539-542
numerice, tipuri de date, 103-104, 517
BIGINT, 518
DECIMAL, 519
DOUBLE, 519
FLOAT, 518
INT, 518
MEDIUMINT, 518
SMALLINT, 517
TINYINT, 517
numerice, valori, 100
coloane, 106
atributul AUTO_INCREMENT, 110-114
atributul NULL, 111
atributul UNSIGNED, 111
atributul ZEROFILL, 110
domenii, 106-107
repartizare în, 107
secvențe, 116
conversie de tip, 152
cu virgulă mobilă, 106, 107
trunchierea valorilor, 109
întregi, 106-107
coloane, 103
trunchiere, 109

O

-O, opțiune de program 604, 607, 613, 624
obiecte (Perl DBI), 294-295
obținere
baza de date demonstrativă, 38
distribuția samp_db, 515
pachete, 501
OCT(), funcție șir, 547
OCTET_LENGTH(), funcție șir, 547
ODBC (Open Database Connectivity)
asistență (instrumente create de terțe părți), 702
MyODBC, instalare, 514
--old-protocol, opțiune de program, 624
--old-server, opțiune de program, 624
omiterea argumentelor legătură din apelurile PHP, 377-378
ON, clauză
instrucțiunea GRANT, 575
instrucțiunea SELECT, 180
--one-database, opțiune de program, 613
--one-thread, opțiune de program, 624
Open Database Connectivity. *Vezi* ODBC
operator de deplasare la stânga (<<), operator pe bit, 533
operator de deplasare spre dreapta (>>), operator la nivel de bit, 532
operator de egalitate care acceptă valori NULL (<=>), 529
operator de inegalitate (!=), operator de comparație, 529
operatori, 525-526
aritmetici, 146, 527-528
comparație, 528-531
sintaxă, 146-149
conversie, 533
echivalențe cu un model, 149, 150, 534-537
grupare, 527
logici, 532-533
sintaxă, 146
pe bit, 533
sintaxă, 146
precedență, 150, 526
operatori aritmetici, 66, 146, 527-528
operatori de comparație, 66, 528-531
identificarea echivalențelor cu un model, 149, 150
sintaxă, 146-149, 150
valoarea NULL, 67-68
operatori de conversie, 533
operatori logici, 66, 532-533
AND (ȘI), 66
OR (SAU), 66
sintaxă, 146
operatori pe bit, 533
sintaxă, 146
operații de golire, executare, 454
optimizare
isamchk, 495-496
myisamchk, 495-496
performanța serverului MySQL, 463-465
optimizarea interogărilor, 195
aspecte hardware, 215
aspecte legate de planificare și blocare, 211-214
indexuri, 195-196
avantaje, 196-199
dezavantaje, 199
selectarea coloanelor pentru, 199-201
încărcarea datelor, 209-211
parametrii serverului, 214
selectarea tipurilor de coloane, 206-208
utilitarul de optimizare a interogărilor MySQL, 201-202
anulare, 205-206
trecere în revistă a operațiilor, 202-205
OPTIMIZE TABLE, instrucțiune, 208, 582
opțiuni
clientul mysql
opțiuni specifice, 612, 614
opțiuni standard, 612
variabile, 614
linie de comandă, 39-40
pentru programe, 598-599
fișiere cu opțiuni, 601-603
opțiuni standard, 599-601
variabile de mediu, 603-604
scriptul mysqlaccess
opțiuni specifice, 616-617
opțiuni standard, 616
serverul mysqld
opțiuni specifice, 622-625
opțiuni standard, 621
utilitarele isamchk și myisamchk
opțiuni specifice acceptate, 605-607
opțiuni standard acceptate, 605
variabile, 608-609
utilitarele myisampack și pack_isam
opțiuni specifice, 610
opțiuni standard, 609
utilitarul mysqladmin
opțiuni specifice, 618
opțiuni standard, 168
utilitarul mysqldump, 459-460
opțiuni privind formatul de date, 630
opțiuni specifice, 628-630
opțiuni standard, 628
utilitarul mysqlimport
opțiuni privind formatul de date, 632
opțiuni specifice, 631-632
opțiuni standard, 631
utilitarul mysqlshow
opțiuni specifice, 633
opțiuni standard, 633
opțiuni de conectivitate, selectarea ISP, 709-710
opțiuni de formatare a datelor
utilitarul mysqldump, 631
utilitarul mysqlimport, 632
opțiuni de program specifice sistemului Windows (serverul mysqld), 625
opțiuni de program standard, listă de, 599-601
opțiuni specifice
clientul mysql, 612, 614
scriptul mysqlaccess, 616-617
serverul mysqld, 622-625
utilitarele isamchk și myisamchk, 605-607
utilitarele myisampack și pack_isam, 610
utilitarul mysqladmin, 618
utilitarul mysqldump, 628-630
utilitarul mysqlimport, 631-632
utilitarul mysqlshow, 633
opțiuni standard
clientul mysql, 612
scriptul mysqlaccess, 616-617
serverul mysqld, 621-622
utilitarele isamchk și myisamchk, 605-606
utilitarele myisampack și pack_isam, 609-610
utilitarul mysqladmin, 617
utilitarul mysqldump, 628
utilitarul mysqlimport, 631
utilitarul mysqlshow, 633
OR, operator, 66
la nivel de bit, 532
logic, 533
ORDER BY, clauză (instrucțiunea SELECT), 69, 79, 585
ordine aleatoare, sortarea rezultatelor în, 206
ordine ascendentă, sortarea rezultatelor înregistrării, 69
ordine de sortare, specificare, 187
organizarea înregistrărilor, 27

P

\p, comandă (clientul MySQL), 615
-p, opțiune, introducerea parolelor, 39-40
-P, opțiune de program, 599, 610, 617
pachete
instalare, 501
liste de discuții, înscriere la, 502
versiune, selectare, 502
pack_isam, utilitar, 597, 609
opțiuni specifice acceptate, 610
opțiuni standard acceptate, 609
utilizare, 609

PACK_KEYS, cuvânt cheie, 570
--packlength, opțiune de program, 610
pagini (Web), generare de la scripturi PHP, 373-374
pagini Web
generare din scripturi PHP, 373-374
transferul parametrilor către modulul CGI.pm, 358-360
param(), metodă (modulul CGI.pm), 356
parametri. *Vezi și* parametri server
conexiuni
specificare, 44, 91
specificare în scripturi Perl, 319-322
stocare, 91-92
script de rotație a jurnalelor generale, 454
transferul de la paginile Web (modulul CGI.pm), 358-360
parametri de asociere (scripturi Perl), 318-319
parametri de conexiune, specificare în scripturi Perl, 319-322
parametri de conexiune la rulare, programe client pentru, 253-266
parametri de intrare (modulul CGI.pm), 356-357
parametrii conexiunii la server, specificare în scripturi Perl, 319-322
parametrii serverului (optimizarea interogărilor), 214
paranteze ( )
funcții, 42
gruparea operatorilor, 527
parole, 39
funcția load_defaults(), 257
introducere, 39-40
stocarea în tabelele user, 479
utilizator rădăcină, configurare, 437
--password, opțiune de program, 599
Password, coloană (coloane de valabilitate), 479
password, comandă (utilitarul mysqladmin), 620
PASSWORD, cuvânt cheie, 570, 586
PASSWORD(), funcție, 479, 484, 562
păstrarea automată a evidenței rezultatelor școlare, 32
păstrarea electronică a evidenței, 32
performanță. *Vezi și* optimizarea interogărilor
factori de selectare a interfețelor, 230-233
isamchk, 495-496
myisamchk, 495-496
performanța serverului MySQL, 463-465
selectarea tipului coloanei, efect asupra, 139-140
structura catalogului de date, efect asupra, 426-428
performanțe de sistem. *Vezi* performanță
PERIOD_ADD(), funcție, 555
PERIOD_DIFF(), funcție, 555
permisiuni
atribuire în cataloage, 471-472
tabele de acordare, precauții, 482-483
PHP Base Library (PHPLIB), 235
PHP/FI, 224
PI(), funcție numerică, 541
--pid-file, opțiune de program, 624
PID (identificator de proces), fișier, 428
ping, comandă (utilitarul mysqladmin), 620
ping(), metodă (Perl DBI), 668
--pipe, opțiune de program, 599
--plan, opțiune de program, 617
planificarea bazelor de date (tabele), 44-47
planificarea întreținerii preventive, 496
plasarea fișierelor PHP include, 381-382
platforme, instalare
UNIX, 503-512
Windows, 512-515
pointeri de buffer (neinițializați), depanare, 292
pointeri de buffer neinițializați, depanare, 292
pointeri neinițializați pentru variabilele de tratare a conexiunilor, 290-291
politică de blocare, 211-214
politică de planificare, 211-214
pornire, 40
mai multe servere MySQL, 466
opțiuni, configurare, 442
script, modificare, 442
script de verificare a tabelelor, rulare, 498
serverul MySQL, 438, 440
verificarea tabelelor, 442
--port, opțiune de program, 600
portabilitate, 115
factori de selectare a interfețelor API, 235-236
interfața API pentru Perl DBI, 224-226
posibilitate de înregistrare-revenire, 189-192
POSITION(), funcție șir, 547
potențial de dezvoltare, selectarea ISP, 711
POW(), funcție numerică, 541
POWER(), funcție numerică, 541
precauții (tabele de acordare), 482-483
precedența operatorilor, 150, 526
PRECISION, atribut (Perl DBI), 676
prefixe (coloane), crearea indexurilor, 171-172
prefixe de stânga, 200
prefixurile coloanelor (crearea indexurilor), 171-172
prelucrarea interactivă a interogărilor, program client pentru, 283-284
prelucrarea interogărilor
API C, 266-268
abordări alternative, 275-276
calcule cu seturi de rezultate, 286-288
care nu returnează seturi de rezultate, 268-269
codarea datelor problematice, 286-288
comparație între mysql_store_result() și mysql_use_result(), 276-278
încărcarea datelor imagine, 288-289
metadatele setului de rezultate, 278-283
rutină de uz general pentru tratarea interogărilor, 272-274
set de rezultate returnat, 269-272
structura cu informații despre tabel, 289
API PHP, 386-390, 687-670
scripturi Perl
asocierea parametrilor, 318-319
câmpuri de înlocuire, 316-319
cicluri de preluare a rândurilor, 305-308
citarea șirurilor, 314-316
fără seturi de rezultate, 304-305
metadatele setului de rezultate, 325-330
număr de rânduri returnate, 308
parametri de conexiune, 319-322
rezultate cu un singur rând, 380-310
seturi de rezultate complete, 310-313
valori NULL, 313-314
prelucrarea punctajelor elevilor (proiectul de păstrare a evidenței rezultatelor școlare), 401-402
prepare(), metodă (Perl DBI), 298, 668
prescurtarea interogărilor, 93-94
preț, selectare ISP, 707-709
--preview, opțiune de program, 617
PRIMARY KEY, cuvânt cheie, 569
PRIMARY KEY, indexuri, 170
--print-defaults, opțiune de program, 600
print, comandă (clientul MySQL), 615
PrintError, atribut (Perl DBI), 300, 675
PRIVILEGES, cuvânt cheie, 574
privilegii
acces de la distanță, acordare, 447
acces la citire, 470
accesul interogării, verificare, 480-481
actualizare, 484-485
administrative, acordare, 448-449
aplicarea la mai multe conturi de utilizator, 449-450
combinare, 481
acordare, 448-449
de acordare a accesului, acordare, 451
FILE, 476
GRANT, 476, 574-576
la nivel de înregistrare, 194
PROCESS, 477
RELOAD, 477
revocare, 451
REVOKE, 582-583
scriptul mysqlaccess, 616
opțiuni specifice acceptate, 616-617
opțiuni standard acceptate, 616
utilizare, 616
selectarea pentru conturile de utilizator, 448-449
SHOW GRANTS, 589
SHUTDOWN, 477
specifice bazelor de date, acordare, 449-450
specifice coloanelor, acordare, 449-450
specifice tabelelor, acordare, 449-450
tabele de acordare, 475-477
privilegii administrative, 475-477
acordare, 448-449
privilegii de acordare a accesului, acordare, 450
privilegii globale, 448-449
privilegii la nivel de bază de date, 481
acordare, 448-449, 484-485
privilegii la nivel de coloană
acordare, 449-450
configurare, 484-485
privilegii la nivel de înregistrare, 194
privilegii la nivel de tabel, configurare, 484-485
PROCEDURE, clauză (instrucțiunea SELECT), 585
PROCEDURE ANALYSE(), funcție, 135, 207
proceduri stocate, 193-194
proces de înștiințare privind plata cotizațiilor (exemplul cu Liga istorică), automatizare, 337-343
PROCESS, privilegiu, 477, 575
process_result_set(), funcție, 270
processlist, comandă (utilitarul mysqladmin), 620
produs cartezian. *Vezi* uniri complete
program modular de conectare cu serverul, program client pentru, 246-253
programare pentru MySQL, avantaje, 219-220
programe. *Vezi și* aplicații; scripturi; utilitare
mysqladmin, configurarea parolelor, 437
mysqldump, 456
opțiuni, 459-460
transferuri în fișiere text, 457-458
opțiuni, 598-599
fișiere cu opțiuni, 601-603
opțiuni standard, 599-601
variabile de mediu, 603-604
programe client. *Vezi* programe client
utilitarul de conversie a numelor, nepotriviri ale numelor utilizatorilor, 448-449
programe client, 36
acces de la distanță, 36
accesul la catalogul de date, 422-423
cerințe de instalare, 37-38
cerințe de sistem, 238-239
cod de conexiune modular, 246-253
compilare și legare, 239-240
conexiuni cu serverul, 240-242
depistarea exemplelor de, 238
mysql, comparație cu MySQL, 38
parametri de conexiune la rulare, 253-266
prelucrarea interactivă a interogărilor, 283-284
verificarea erorilor, 242-246
programe de acces la client, selectare ISP, 709
programe de calcul tabelar, comparație cu sistemele de baze de date, 54-55
proiectarea tabelelor dintr-o bază de date, 44-47
proiectul de evidență a rezultatelor școlare
bază de date demonstrativă, 31-32
browser de punctaje, 366-369
tabele, 50
absențe, 59
creare, 57
elev, 55-56
eveniment, 52-55, 58
puncte, 50-51, 59
prompt de interpretor, 39
prompturi
modul interactiv al programului client mysql, 611
simbolul procent (%), 38
proprietăți
funcția mysql_fetch_field(), 690
funcția mysql_field_flags(), 696-697
protecția fișierelor, 470
protocoale de blocare, 492-494
ps, comandă, 421, 444
punct și virgulă (;), comandă, clientul mysql, 615

Q

\q, comandă (clientul mysql), 615
-q, opțiune de program, 606, 613, 629
qq{}, construcție (Perl DBI), 314-315
QUARTER(), funcție, 556
--quick, opțiune de program, 459, 606, 613, 629
QUIT, comandă, 41
quit, comandă (clientul mysql), 615
quote(), metodă (Perl DBI), 316, 668

R

\r, comandă (clientul mysql), 615
-r, opțiune de program, 607, 613, 632
RADIANS(), funcție numerică, 541
RaiseError, atribut (Perl DBI), 300-302, 675
RAND(), funcție numerică, 541
rapoarte de semnalare a hibelor (scriptul mysqlbug), 620-621
raportarea erorilor (PHP), dezactivare, 391
--raw, opțiune de program, 613
rânduri
  inserție, 60
  lungime, selectare, 118
  sortare, 68-69
  ștergerea lentă a conținutului tabelelor, 205
rânduri de lungime fixă
  conversia la rânduri de lungime variabilă, 175
  conversia rândurilor de lungime variabilă la, 174-175
  șiruri CHAR, 118
rânduri de lungime variabilă
  conversia în rânduri de lungime variabilă, 175
  conversia rândurilor de lungime fixă în, 175
  șiruri VARCHAR, 118
--read-only, opțiune de program, 608
reclame pe manșete, 28
--recover, opțiune de program, 607
redenumirea tabelelor, 174
redirecționare, 42
redirecționarea datelor de intrare, 42
reducere
  fragmentare, 120
  timpul de nefuncționare a serverului MySQL, 495
refacere în urma căderilor, 418
refacerea datelor sistemului, 461-462
refacerea în urma erorilor, 418
REFERENCES, clauză (instrucțiunea CREATE TABLE), 164
REFERENCES, privilegiu, 476, 575
referințe (Perl DBI), 294-295
referirea la coloane în expresii, 146
refresh, comandă (utilitarul mysqladmin), 620
regăsire
  date, 72-74
  date din tabele, 62
  înregistrări, 27
    din tabele multiple, 82-88
    valoarea NULL, 67-68
REGEXP, operator de identificare a echivalențelor cu un model, 535
reglarea serverului MySQL, 463-465
rehash, comandă (clientul mysql), 615
reîncărcarea tabelului de acordare, 438, 451
--relative, opțiune de program, 618
relații cu clienții, selectare ISP, 707-708
„relațional", cuvânt, definiție, 33
RELEASE_LOCK(), funcție, 562
--relnotes, opțiune de program, 617
reload, comandă (utilitarul mysqladmin), 620
RELOAD, privilegiu, 477, 575
reluarea comenzilor, 92
--remove, opțiune de program, 625
reparare standard a tabelelor, executare, 491
repararea tabelelor, 470. *Vezi și* depanare
  cerințe, 488-489
  protocoale de blocare, 494-495
  reparare standard, 491
  utilitarele isamchk și myisamchk, 604-609
repararea tabelelor, 490-491
  metoda standard, 491
  protocoale de blocare, 494-495
REPEAT(), funcție șir, 547
--replace, opțiune de program, 632
REPLACE, instrucțiune, 571, 582
REPLACE(), funcție șir, 547
replicarea bazelor de date, 461
repornirea serverului MySQL, 443-444
reputație, selectare ISP, 711
require(), funcție, 382
resecvențierea coloanelor, 187
restricții (SQL), 220
restricționare
  accesul utilizatorilor, 447
  rezultatele interogării, 69
    clauza HAVING, 80
rețele
  accesul clienților, control, 477-478
  securitate, 472-473
    tabele de acordare, 473-475
  servere MySQL
    multiple, rulare, 465
    oprire-pornire, configurare, 438
    timp de nefuncționare, reducere, 495
revenire, capacitate de, 189-192
REVERSE(), funcție șir, 548
revocarea privilegiilor, 451
REVOKE, instrucțiune, 582-583
  efectul asupra tabelelor de acordare, 444-445
  sintaxă, 451
rezoluție de nume, depanare, 448-449
rezolvarea numelor complet determinate, 448-449
rezultate (ale interogărilor)
  calcul, 71
  eliberarea memoriei alocate, 378
  identificarea echivalențelor cu un model, 75-76
  limitare, 70
  PHP, prelucrare, 387-388
  restricționare (clauza HAVING), 80
  rezumare, 76-82
  sortare, 68-69
    clauza GROUP BY, 79
rezultate cu un singur rând, preluare cu scripturi Perl, 380-310
rezumare
  coloane multiple, 86
  informații, 27
  înregistrări, 76-82
--rhost, opțiune de program, 617
Rich Text Format (RTF), 330-331
  catalogul Ligii istorice, exemplu, 335-337
RIGHT(), funcție șir, 548
RLIKE, operator de identificare a corespondenței cu un model, 536
--rollback, opțiune de program, 617
rotație
  fișiere jurnal, 453
    sisteme Linux, 454
  jurnale de actualizare, 454-455
  script pentru jurnal general, rotire, 454
ROUND(), funcție numerică, 542
rows(), metodă (Perl DBI), 672
RPAD(), funcție șir, 548
RPM, distribuții (sisteme UNIX), instalare, 505-507

RTF (Rich Text Format), 31, 330-331
  catalogul Ligii istorice, exemplu, 335-337
RTRIM(), funcție șir, 548
rularea de servere MySQL multiple, 465
rutină de uz general pentru tratarea interogărilor, 272-274
rutine. *Vezi* funcții; metode

S-Ș

\s, comandă (client MySQL), 615
-S, opțiune de program, 600, 607
--safe-mode, opțiune de program, 624
--safe-recover, opțiune de program, 607
safe_mysqld, script, 598, 633
  invocare, 440-441
samp_db, distribuție, instalare, 515
SCALE, atribut (Perl DBI), 677
scriere
  cicluri de preluare a rândurilor cu scripturi Perl, 305-308
  expresii, 145
    conversie de tip, 152-155
    operatori, 145-149, 150
    referințe la coloane, 146
    valori NULL, 151
  scripturi de interpretor, 96
  scripturi Perl, 664
  scripturi PHP, 680-681
    delimitatori, 373-374
SCRIPT, tip de date, 375
scripturi, 73. *Vezi și* aplicații; programe; utilitare
  bazate pe Web, 353-354
    browser demonstrativ pentru baze de date, 362-366
    browser pentru interese comune (exemplul cu Liga istorică), 370-371
    browser pentru punctaje din proiectul pentru evidența rezultatelor școlare, 366-369
    conectarea la serverul MySQL, 360-361
    configurarea serverului Apache, 354-355
    scripturi CGI.pm, 355-360
  coloane, completare, 115
  de pornire, modificare, 442
  expirare, rulare, 455
  Liga istorică
    editarea on-line a intrărilor, 405-412
    ghicitoarea cu președinții americani, 401-405
mysql.server, 597, 616
  invocare, 440-441
mysqlaccess, 597, 616
  opțiuni specifice acceptate, 616-617
  opțiuni standard acceptate, 616
  utilizare, 616
mysqlbug, 597, 620-621
operații de golire, execuție, 454
Perl, 294
  asocierea parametrilor, 318-319
  câmpuri de înlocuire, 316-319
  cicluri de preluare a rândurilor, 305-308
  citarea șirurilor, 314-316
  depanare, 322
  depanare cu parametri de afișare, 322-323
  depanare cu urmărire, 324-325
  exemplu, 295-300
  metadatele setului de rezultate, 325-330
  număr de rânduri returnate, 308
  parametri de conexiune, 319-322
  prelucrarea înregistrărilor fără seturi de rezultate, 304-305
  rezultate cu un singur rând, 380-310
  scriere, 664
  seturi de rezultate complete, 310-313
  tratarea erorilor, 300-303
  valori NULL, 313-314
PHP
  caracteristici, 373
  comparație cu scripturi DBI, 379
  eliminarea mesajelor de eroare, 377-378
  etichete, 375
  exemplu, 375-376
  fișiere include, 380-384
  pagini Web, generare, 373-374
  scriere, 680-681
  selecția bazei de date, 378
  tratarea erorilor, 385
rotația jurnalelor generale, 454
safe_mysqld, 598, 633
  invocare, 440-441
scriere, 96
verificarea tabelelor
  executare, 496-497
  executare la pornire, 498
  invocare din fișierul crontab, 497
scripturi bazate pe Web, 353-354
  browser demonstrativ pentru baze de date, 362-366
  browser pentru interese comune (exemplul cu Liga istorică), 370-371
  browser pentru punctaje din proiectul pentru evidența rezultatelor școlare, 366-369
  conectarea la serverul MySQL, 360-361
  configurarea serverului Apache, 354-355
  scripturi CGI.pm, 355-360
scripturi CGI, configurarea serverului Apache pentru, 354-355
scripturi PHP autonome, 375
scurtături (interogări), 93-94
SEC_TO_TIME(), funcție, 556
SECOND(), funcție, 556
securitate
  acces la citire, 470
  accesul clienților, 477-478
  atribuții administrative, 417
  catalogul de date, 431
  coloane pentru domeniul de valabilitate, 478-479
    Column name, 479
    Db, 479
    Host, 478
    Password, 479
    Table name, 479
    User, 479
  combinarea tabelelor, 426
  consemnare în (nume de utilizatori), 39
  copii de siguranță de sistem, 457-458
    copiere directă, 460
    refacerea datelor, 461-462
    transferul în fișiere text, 457-458
  externă, 472-473
    tabele de acordare, 473-475
  fișiere jurnal, 470
  parole, 39
    introducere, 39-40
    stocare în tabelele user, 479
  permisiuni, atribuire pentru cataloage, 471-472
  privilegii
    combinare, 481
    FILE, 476
    GRANT, 476
    la nivel de baze de date, 481
    PROCESS, 477
    RELOAD, 477
    revocare, 451

SHUTDOWN, 477
scripturi bazate pe Web, 360-361
selectare ISP, 711, 713
tabele de acordare
compatibilitate înapoi, 483-484
internă, 470
load_defaults(), funcție, 257
măsuri de precauție, 482-483
numele gazdei locale, specificare, 448-449
privilegii administrative, 475-477
protocoale de blocare, 492-494
reîncărcare, 438, 451
sensibilitate la diferența între majuscule și minuscule, 479-480
utilizatori anonimi, 437
securitate externă, 472-473
accesul clientului, 477-478
tabele de acordare
coloane cu privilegii, 473-474
coloane pentru domeniul de valabilitate, 473-474
conținut, 473-474
privilegii, 475
structură, 473-474
securitate internă, 470
secvențe, 112
generare, 116
numere unice, generare, 112-113
valoare inițială, specificare, 115
valori negative, 114
secvențe de expresii regulate extinse, 535
SELECT, instrucțiune, 62, 177, 583-586
clauze, 63
crearea tabelelor din rezultate, 167-169
GROUP BY, clauză, 79
HAVING, clauză, 80
LIMIT, clauză, 70
ORDER BY, clauză, 69
uniri complete, 178-179
uniri la stânga, 180-181
uniri normale, 178
WHERE, clauză, 64-65
SELECT, privilegiu, 476, 575
selectall_arrayref(), metodă (Perl DBI), 312, 669
selectare
baze de date
prestabilite, 163
script PHP, 378
instrumente pentru instalare, 500-501
interfețe API, 228
mediu de execuție, 229
performanțe, 230-233
portabilitate, 235-236
timp de dezvoltare a programului, 233-235
ISP, 704-705
considerații privind lărgimea de bandă, 706
criterii generale, 707-711
criterii specifice sistemului MySQL, 712-714
determinarea necesităților dumneavoastră, 705-706
servicii furnizate, 706
MySQL, avantajele, 13-14
opțiuni de pornire, 442
pachete, versiune, 502
privilegii pentru conturi de utilizator, 448-449
sistemul de operare, 31
tipuri de coloane, 133-138
aspecte de inter-relaționare, 143
aspecte de performanță, 139-140
compararea coloanelor de lungimi fixe și variabile, 140-141
compararea dimensiunii tipurilor, 140
domenii de valori, 138-139
efectul asupra comparației valorilor, 142
posibilitate de indexare, 142-144
valori NULL, 141
selectcol_arrayref(), metodă (Perl DBI), 669
selectrow_array(), metodă (Perl DBI), 309-310, 669
semne ale citării (ghilimele duble), 101
semnul egal (=), operator de comparație, 529
semnul exclamării (!), operator logic, 533
semnul minus (-), operator aritmetic, 527
semnul plus (+), operator aritmetic, 527
sensibilitate la diferența între majuscule și minuscule
comparații între șiruri, 529
instrucțiuni SQL, 162
numele tabelelor, 64
tabele de acordare, 479-480
separare
argumentele funcțiilor, 145
tipuri de date, 52-53
server Apache
configurarea pentru scripturile CGI, 354-355
instalare pe UNIX, 510-512
instalare pe Windows, 514-515
server MySQL
actualizare, 467
conexiuni prin scripturi bazate pe Web, 360-361
conturi de utilizator, gestionare, 444-445
fus orar, configurare, 442
interacțiunea cu utilitarele, 492
multiple, configurare, 466
oprire, 442-443
pornire, 440
opțiuni, 442
servere multiple, 466
verificarea tabelelor, 442
repornire, 443-444
terminare, 444
servere, 36
clienți, 36
acces de la distanță, 36
conexiuni, 39-41, 437
funcția mysql_pconnect(), 376-378
închidere, 41
multiple, 417
MySQL
actualizare, 467
ajustare, 463-465
configurarea serverelor multiple, 466
conturi de utilizator, gestionare, 444-445
interacțiunea cu utilitarele, 492
multiple, pornire, 466
numele de gazdă locală, specificare în tabelul de acordare, 448-449
oprire, 442-443
pornire, 438, 440, 442
repornire, 443-444
rularea cu un cont de utilizator fără privilegii, 438-440
rularea serverelor multiple, 465
terminare, 444
timp de nefuncționare, reducere, 495
variabile, 463
mysqld, 415
accesul la catalogul de date, 422-423
atribuții generale de administrare, 416-417
securitate, 417
Web, regăsirea reclamelor, 28
servere multiple, 417
servicii furnizate, selectarea ISP, 706
sesiuni, terminare, 41
SESSION_USER(), funcție, 562

--set_auto_increment(), opțiune de program, 608
--set-variable, opțiune de program, 600
SET, clauză (instrucțiunea INSERT), 577
SET, instrucțiune, 586-588
SET, tip de date, 104, 121-125, 523
comparație cu ENUM, 124
indexare, 125
valori admise, 125
set de rezultate ratat, depanare, 291
seturi de rezultate
calcule cu (prelucrarea înregistrărilor), 284-286
inexistente, prelucrarea interogărilor, 268-269
metadate, prelucrarea înregistrărilor, 278-283
prelucrarea cu scripturile Perl
cicluri de preluare a rândurilor, 305-308
fără seturi de rezultate, 304-305
metadate, 325-330
număr de rânduri returnate, 308
rezultate cu un singur rând, 380-310
seturi de rezultate complete, 310-313
valori NULL, 313-314
ratate, depanare, 291
returnate, prelucrarea interogărilor, 269-272
seturi de rezultate complete, prelucrare cu scripturi Perl, 310-313
-Sg, opțiune de program, 624
SGBDR (sistem de gestiune a bazelor de date relaționale), 33
comparație cu programele de calcul tabelar, 54-55
SHOW, instrucțiune, 588
SHOW COLUMNS, instrucțiune, 589
SHOW COLUMNS FROM, instrucțiune, 176
SHOW DATABASES, instrucțiune, 50, 176, 424, 589
SHOW FIELDS, instrucțiune. *Vezi* **SHOW COLUMNS, instrucțiune**
SHOW FIELDS FROM, instrucțiune, 289
SHOW GRANTS, instrucțiune, 589
SHOW INDEX FROM, 176
SHOW KEYS, instrucțiune. *Vezi* **SHOW INDEX, instrucțiune**
SHOW PROCESSLIST, instrucțiune, 590-591
SHOW STATUS, instrucțiune, 591-593
SHOW TABLE STATUS, instrucțiune, 176, 593-594
SHOW TABLES, instrucțiune, 50, 176, 424, 594
SHOW TABLES FROM, instrucțiune, 176
SHOW VARIABLES, instrucțiune, 594
shutdown, comandă (utilitarul mysqladmin), 620
SHUTDOWN, privilegiu, 477, 575
SIGN(), funcție numerică, 542
--silent, opțiune de program, 600
SIN(), funcție numerică, 542
sintaxă
configurarea variabilei TZ, 442
configurarea variabilelor (serverul MySQL), 463
declararea coloanelor, 104
domeniile coloanelor de tip numeric, 106-107
instrucțiunea ALTER TABLE, 90
instrucțiunea CREATE TABLE, 47
instrucțiunea GRANT, 445-457
instrucțiunea REVOKE, 451
instrucțiunea SELECT, 63
invocarea funcțiilor, 145
invocarea utilitarului myisamchk, 489-490
operatori de comparație, 147-149, 150
operatori logici, 146
operatori pe bit, 146-147
PHP, influențe lingvistice, 380
sistem de fișiere
copii de siguranță, 457-458
securitate, 417
securitate internă, 470
sistem de gestiune a bazei de date demonstrative 8-10
sisteme de gestiune, 33
sisteme de gestiune a bazelor de date, 24-25, 26
acces de la distanță, 28
demonstrative, 28-31
păstrarea evidenței rezultatelor școlare, 32
sisteme de operare, selecție, 31
sisteme de gestiune a programelor, 33
sisteme de operare
aspecte privind licența pentru MySQL, 16
efect asupra sensibilității la diferența între majuscule și minuscule, 64
Linux, rotația fișierelor jurnal, 454
restricții asupra convențiilor de nume, 425-426
restricții asupra dimensiunii fișierelor, 427-428
selecție, 31
UNIX
Apache, 510-512
conturi de utilizator, creare, 504
distribuție, instalare, 504-507
distribuții, obținere, 504-505
inițializarea bazei de date mysql, 507
PHP, 510-512
suportul Perl DBI, 509-510
Windows
Apache, 515
distribuția client-server, 513-514
MyODBC, 514
PHP, 515
suportul Perl DBI, 514-515
situri (Web)
pachete, 501
reclame pe mașete, 28
situri oglindă (MySQL), 501
situri Web
pachete, 501
reclame pe mașete, 28
--skip-column-names, opțiune de program, 613
--skip-grant-tables, opțiune de program, 624
--skip-host-cache, opțiune de program, 624
--skip-line-numbers, opțiune de program, 613
--skip-locking, opțiune de program, 624
--skip-name-resolve, opțiune de program, 624
--skip-networking, opțiune de program, 624
--skip-new-program, opțiune de program, 624
--skip-show-database, opțiune de program, 625
--skip-thread-priority, opțiune de program, 625
slash(/), operator aritmetic, 527
--sleep, opțiune de program 608
SMALLINT, tip de date, 103, 106-107, 517

--socket, opțiune de program, 600
Solaris, comenzi de pornire, 442
solicit_event_info(), funcție, 399
--sort-index, opțiune de program, 607
--sort-records, opțiune de program, 607
sortare
coloane, 69
coloane șir, SET și ENUM, 125
înregistrări, 27
valori NULL, 69
rezultate în ordine aleatoare, 206
rezultatele interogării, 68-69
clauza GROUP BY, 79
ordine crescătoare, 69
SOUNDEX(), funcție șir, 548
SPACE(), funcție șir, 548
--spassword, opțiune de program, 617
specificare
atributele coloanelor, 105
criterii de regăsire, 64-67
expresii, 71
identificarea echivalențelor cu un model, 75-76
valori NULL, 67-68
numele gazdei locale în tabelul de acordare, 448-449
opțiuni de pornire, 442
parametri de conexiune, 44, 91
rezultatele interogării (date calendaristice), 72-74
valoarea inițială în secvențe, 115
specificatori ai tipului de stocare (instrucțiunile CREATE TABLE), 165-166
SQL (Structured Query Language), 35
administrarea sistemului MySQL, 416-417
caracteristici acceptate-inexistente în MySQL, 158-160
instrucțiuni. *Vezi* instrucțiuni (SQL)
restricții, 220
SQL_AUTO_IS_NULL, cuvânt cheie, 587
SQL_BIG_RESULT, cuvânt cheie, 583
SQL_BIG_SELECTS, cuvânt cheie, 587
SQL_BIG_TABLES, cuvânt cheie, 587
SQL_LOG_OFF, cuvânt cheie, 587
SQL_LOG_UPDATE, cuvânt cheie, 587
SQL_LOW_PRIORITY_UPDATES, cuvânt cheie, 587
SQL_SELECT_LIMIT, cuvânt cheie, 588
SQL_SMALL_RESULT, cuvânt cheie, 583
SQL_WARNINGS, cuvânt cheie, 588
SQRT(), funcție numerică, 542
stabilirea conexiunilor cu serverul, 39-41, 376-378
--standalone, opțiune de program, 625
STATEMENT, atribut (Perl DBI), 677
--status, opțiune de program, 633
status, comandă
clientul MySQL, 616
utilitarul mysqladmin, 620
STATUS, cuvânt cheie, 574
STD(), funcție de sumar, 559
STDDEV(), funcție de sumar, 559
stil compatibil ASP (scripturi PHP), 375
stil de etichetă prestabilit (scripturi PHP), 375
stiluri, etichete PHP, 375
stiluri de etichetă (HTML)
generare, 383
PHP, 375, 680-681
stocare
fișierele PHP include, 381-382
interogări, 41
parametri de conexiune, 91-92
parole în tabelul user, 480
STRAIGHT_JOIN, cuvânt cheie, 179, 584
STRCMP(), funcție de comparație, 538
structură
catalog de date, 421-422
tabele
afișare, 289
modificare, 90
tabele de acordare, 473-474
Structured Query Language. *Vezi* SQL
SUBDATE(), funcție, 556
subselecții, 189
rescriere sub formă de uniri, 182-183
SUBSTRING(), funcție șir, 549
SUM(), funcție, 80-81, 559
--superuser, opțiune de program, 617
super-utilizatori. *Vezi și* utilizator rădăcină
creare, 448-449
privilegii, acordare, 484-485
System V, instalarea aplicației mysql.server, 440-441
SYSTEM_USER(), funcție, 563
șiruri, 101
cerințe privind spațiul de stocare, 117
coloane, 103-104, 117-120
ENUM, 121-125
selectare, 133-144
SET, 121-125
TEXT, 120
valori BLOB, 120
comparație, 149-150
funcții de construcție a interogărilor (API C), 646-647
ghilimele duble, 101
operatori de identificare a echivalențelor cu un model, 534-537
prelucrarea interogărilor, codificarea datelor problematice, 286-288
secvențe escape, 537
semnele citării în scripturile Perl, 314-316
valori caracter cu lungime variabilă, repartizare în coloane, 48
ștergere
conturile de utilizator din tabelul user, 451
intrări din tabelul user, 88
înregistrări duplicate, 76-77
lentă a conținutului unui tabel, 205
ștergere
configurații stocate (utilitar de configurare), 502
interogări cu mai multe linii, 41-42
ștergeri în cascadă, 193-194

T

-T, opțiune de program, 608, 617-618, 623, 625, 629-630
--tab, opțiune de program, 629-630
tabel host (tabel de acordare), 474
tabele
afișarea conținutului, 62
aliasuri, 585
AUTO_INCREMENT, coloană, inserție, 186
chei externe, 192-193
coloane. *Vezi* coloane
comprimate, utilitarele myisampack și pack_isam, 609-610
convenții de nume, 425-426
sensibilitatea la diferența între majuscule și minuscule, 64, 161-162
conversia tipurilor, 176
copiere, 167-168
creare, 44-47, 104
baza de date demonstrativă, 47-49
proiectul de evidență a rezultatelor școlare, 57
cu rânduri de lungime variabilă, conversie la rânduri de lungime fixă, 174-175
date duplicate, 53
date imagine, încărcare, 288-289
deteriorate, detecție, 488-489
executarea copiilor de siguranță, 456, 459
existență, verificare, 188-189
fragmentare, 208
reducere, 120
indexuri. *Vezi* indexuri
informații despre structură, afișare, 289
INSERT, instrucțiune, 576-577
instrucțiunea ALTER TABLE, 90, 173-175, 564-566
instrucțiunea CREATE TABLE, 57, 106, 164-165, 567-571
crearea condiționată a tabelelor, 166
rezultatele interogării SELECT, crearea tabelelor pornind de la, 167-169
specificatori ai tipului de stocare, 165-166
tabele temporare, 167
instrucțiunea DELETE, 571-572
instrucțiunea DESCRIBE, 49, 572
instrucțiunea DROP TABLE, 169, 573
instrucțiunea EXPLAIN, 573-574
instrucțiunea LOAD DATA, 578-581
instrucțiunea LOCK TABLES, 190-191
instrucțiunea OPTIMIZE TABLE, 582
instrucțiunea REPLACE, 582
instrucțiunea SELECT, 177, 583-586
uniri complete, 178-179
uniri la stânga, 180-181
uniri normale, 178
instrucțiunea SHOW COLUMNS, 589
instrucțiunea SHOW TABLE STATUS, 593-594
instrucțiunea SHOW TABLES, 50, 594
ISAM, copiere, 456
înregistrări. *Vezi* înregistrări
întreținere preventivă, planificare, 496
legare, 53
Liga istorică americană, 44
modificarea amplasamentului, 434
modificări ale structurii, 90
nume complet determinate, 161
ordini de sortare, specificare, 187
performanțele sistemului, 426-428
privilegii, acordare, 449-450
privilegii la nivel de înregistrare, 194
proiect de evidență a rezultatelor școlare, 50
tabelul cu absențe, 59
tabelul cu elevi, 55-56
tabelul cu evenimente, 52-55, 58
tabelul cu punctaje, 50-51, 59
protocol de blocare internă, 492-494
rânduri. *Vezi* rânduri
rânduri de lungime fixă, conversie în tabele cu rânduri de lungime variabilă, 175
redenumire, 174
refacerea pe baza copiilor de siguranță, 461-462
regăsirea de informații despre, 176
reparare, 490-491
cerințe, 488-489
protocoale de blocare, 494-495
reparare standard, 491
utilitarele isamchk și myisamchk, 604-609
reprezentare în catalogul de date, 424
subselecții, 189
ștergerea lentă a conținutului, 205
ștergeri în cascadă, 193-194
tabele contor, 187-188
temporare, 567
UNION, operații, 185-186
uniri, 85-88
rescrierea subselecțiilor sub formă de, 182-183
UNLOCK TABLES, instrucțiune, 190-191, 594
UPDATE, instrucțiune, 595
valori de date, identificare, 45
valori inexistente, verificare, 183-185
verificare, 490
protocoale de blocare, 493
scripturi, 496-497
virtuale, 194
tabele comprimate
utilitare pack_isam, opțiuni specifice acceptate, 610
utilitarele myisampack și pack_isam, 609
opțiuni specifice acceptate, 610
opțiuni standard acceptate, 609
utilizare, 609
tabele contor, 187-188
tabele de acordare, 470
coloane de privilegii, 473-474
coloane pentru domeniul de valabilitate, 473-474
conținut, 478-479
ordine de identificare a echivalențelor, 482
compatibilitate înapoi, 483-484
conținut, 473-474
interogări, privilegii de acces, 480-481
măsuri de precauție, 482-483
numele gazdei locale, specificare, 448-449
privilegii, 475
la nivel de bază de date, 481
privilegii administrative, 475-477
reîncărcare, 438, 451
sensibilitate la diferența între majuscule și minuscule, 479
sisteme UNIX, inițializare, 507
structură, 473-474
tabelul user
ștergerea intrărilor, 452
ștergerea înregistrării unui utilizator, 451
tabele deteriorate, detectare, 488-489
tabele temporare, 568
instrucțiuni CREATE TABLE, 167
tabele virtuale, 194
tabelul cu președinți (Liga istorică americană), 45-46
tabelul cu punctaje (proiectul de evidență a rezultatelor școlare), 50-51, 59
tabelul eveniment (proiectul de evidență a rezultatelor școlare), 52-55, 58
tabelul membru (Liga istorică americană), 46
--table, opțiune de program, 613, 617
Table name, coloană (coloane pentru domeniul de valabilitate), 479
table_cache, variabilă, (serverul MySQL), 464

table_priv, tabel (tabel de acordare), 474
TABLES, cuvânt cheie, 574
tablouri (PHP), acces la, 379, 390
TAN(), funcție numerică, 542
taxe, licențe MySQL, 32
Telnet, selectare ISP, 706
TEMPORARY, cuvânt cheie, 567
terminare
instrucțiuni, 42
server MySQL, 444
sesiuni, 40
TERMINATED BY, clauza (instrucțiunea LOAD DATA), 580
terminologie
arhitecturală, 36-37
limbajul interogărilor, 35
structurală, SGBDR (sistem de gestiune a bazelor de date relaționale), 33
terminologie pentru arhitectură, 36-37
terminologie structurală, SGBDR (sistem de gestiune a bazelor de date relaționale), 33
--test, opțiune de program, 610
testarea conversiei de tip, 155-156
text
coloane, șiruri, 117-120
formatare, RTF (Rich Text Format), 31
șiruri, 101
cerințe privind spațiul de stocare, 117
ghilimele duble, 101
TEXT, tip de date, 104, 120, 522
text HTML, modificarea semnificației în modulul CGI.pm, 357
text URL, modificarea semnificației în modulul CGI.pm, 357
thread_id (Perl DBI), 675
TIME, tip de date, 104, 127, 523
TIME_FORMAT(), funcție, 556
TIME_TO_SEC(), funcție, 556
--timeout, opțiune de program, 618
TIMESTAMP, tip de date, 104, 128-129, 524
timp de dezvoltare, factori de selecție a interfețelor, 233-235
timp de dezvoltare a programului, factori de selectare a interfețelor API, 233-235
timp de nefuncționare (serverul MySQL), reducere, 495
TINYBLOB, tip de date, 84, 504. *Vezi și* TEXT, tip de date
TINYINT, tip de date, 103, 106-107, 517
cerințe privind spațiul de stocare, 107
TINYTEXT, tip de date, 84, 505. *Vezi și* TEXT, tip de date
tipuri de coloane. *Vezi* tipuri de date
tipuri de date
API C, 635
macrocomenzi accesoriu, 639
non-scalare, 635-639
scalare, 635
coloane
dată-oră, 103
numerice, 103-110
șiruri, 103-104, 117
convenții de descriere, 516-517
dată-oră, 523
DATE, 523
DATETIME, 524
TIME, 523
TIMESTAMP, 524
YEAR, 524
numerice, 100, 517
BIGINT, 518
cu virgulă mobilă, 107
DECIMAL, 519
DOUBLE, 519
FLOAT, 518
INT, 518
întregi, 106-107
MEDIUMINT, 518
SMALLINT, 517
TINYINT, 517
Perl, 294-295
șir, 101, 519
BLOB, 521
CHAR, 520
ENUM, 522
LONGBLOB, 521
LONGTEXT, 522
MEDIUMBLOB, 521
MEDIUMTEXT, 522
SET, 523
TEXT, 522
TINYBLOB, 520
TINYTEXT, 521
utilizarea ghilimelelor duble, 101
VARCHAR, 520
valori de date, 102
valori de oră, identificare, 102
valori NULL, 102
tipuri de date explicite, 99
tipuri de date implicite, 99
tipuri de date non-scalare (API C), 635-639
tipuri de date scalare (API C), 635
tipuri de date șir, 103, 519
BLOB, 521
CHAR, 520
ENUM, 522
LONGBLOB, 521
LONGTEXT, 522
MEDIUMBLOB, 521
MEDIUMTEXT, 522
SET, 523
TEXT, 522
TINYBLOB, 520
TINYTEXT, 521
VARCHAR, 520
tipuri de fișiere (extensii), 424
tipuri numerice cu dublă precizie, 109
tipuri numerice cu simplă precizie, 109
--tmpdir, opțiune de program, 608, 610, 625
TMPDIR, variabilă de mediu, 604
TO, clauză (instrucțiunea GRANT), 575
TO_DAYS(), funcție de tip dată și oră, 73, 557
trace(), metodă (Perl DBI), 324, 672-673
trace_msg(), metodă (Perl DBI), 672
tranzacții, 189-192
tratarea erorilor
PHP, 391-392
mysql_query(), 389
scripturi Perl, 300-303
TRIM(), funcție șir, 549
TRUNCATE(), funcție numerică, 542
TYPE, atribut (Perl DBI), 677
TYPE, cuvânt cheie, 571
TZ, variabilă, configurare, 442

U

\u, comandă (clientul MySQL), 615
-u, opțiune de program, 600, 607, 617
UCASE(), funcție șir, 550
--unbuffered, opțiune de program 604
UNION, operații, 185-186
UNIQUE, cuvânt cheie, 569
UNIQUE, indexuri, 170
uniri, 82-88
complete, 82-88
la stânga, 180-181
determinarea valorilor inexistente, 183-185
LEFT JOIN, cuvânt cheie, 85
normale, 178
rescrierea subselecțiilor ca, 182-183
SELECT, instrucțiune, 584
uniri complete, 178-179
uniri în trei sensuri, 82
uniri încrucișate. *Vezi* uniri complete
uniri la stânga, 180-181
detectarea valorilor inexistente, 183-185
uniri normale, 178
unități de măsură, afișare, 136-137
unități monetare, afișare, 136
UNIX
aspecte privind licențele, 16
MySQL, instalare, 503
Apache, 510-512
conturi de utilizator, creare, 504
distribuție, instalare, 504-507
distribuții, obținere, 504-505
inițializarea bazei de date mysql, 507
PHP, 510-512
suportul Perl DBI, 509-510
nume de utilizator, 39
prompturi de interpretor, 39
securitatea sistemelor de fișiere, 417
UNIX_TIMESTAMP(), funcție, 557
UNLOCK TABLES, instrucțiune, 190-191, 594
--unpack, opțiune de program, 607
UNSIGNED, atribut (coloane numerice), 111
UPDATE, instrucțiune, 90, 595
UPDATE, privilegiu, 476, 575
UPPER(), funcție șir, 550
urmărire (depanarea scripturilor Perl), 324-325
USAGE, tip de privilegiu, 575
use, comandă (clientul mysql), 615
USE, instrucțiune, 43, 163, 595
use DBI, comandă (Perl), 297
use strict, comandă (Perl), 297
--user, opțiune de program, 600
User, coloană (coloane pentru domeniul de valabilitate), 479
user, tabel (tabel de acordare), 474
înregistrările utilizatorilor, eliminare, 451
parole, stocare, 480
ștergerea intrărilor, 452
USER, variabilă de mediu, 604
USER(), funcție, 563
USING, clauză (instrucțiune SELECT), 180
utilitare, 201-202. *Vezi și* aplicații; programe; scripturi; instrumente create de terțe părți
configurare, 502
instalare, 500-501
interacțiunea cu serverul MySQL, 492
isamchk, 415, 597, 604-605
interacțiunea cu serverul, 492
opțiuni specifice acceptate, 605-607
opțiuni standard acceptate, 605
repararea tabelelor, 491
sintaxă de invocare, 489-490
utilizare, 604-605
variabile, 608-609
verificarea tabelelor, 490
myisamchk, 415, 597, 604-605
interacțiunea cu serverul, 492
opțiuni specifice acceptate, 605-607
opțiuni standard acceptate, 605
repararea tabelelor, 491
sintaxă de invocare, 489-490
utilizare, 604-605
variabile, 608-609
verificarea tabelelor, 490
versiuni viitoare, 415, 597, 604-605
myisampack, 597, 609
myisampack, utilitar, 597, 609
opțiuni specifice acceptate, 610
opțiuni standard acceptate, 609
utilizare, 609
mysql, 415
mysql, client, 597, 611
comenzi, 615
opțiuni specifice acceptate, 612, 614
opțiuni standard acceptate, 612
utilizare, 612
mysqladmin, 415, 597, 617
comenzi, 618-620
opțiuni acceptate specifice, 618
opțiuni acceptate standard, 618
utilizare, 618
mysqldump, 416, 597, 628
opțiuni acceptate privind formatul datelor, 630
opțiuni specifice acceptate, 628-630
opțiuni standard acceptate, 628
utilizare, 628
variabile, 630
mysqlimport, 62, 597, 631
opțiuni acceptate privind formatul datelor, 632
opțiuni specifice acceptate, 631-632
opțiuni standard acceptate, 631
utilizare, 631
mysqlshow, utilitar, 597, 633
opțiuni specifice acceptate, 633
opțiuni standard acceptate, 633
utilizare, 633
pack_isam, 597, 609
opțiuni specifice acceptate, 610
opțiuni standard acceptate, 609
utilizare, 609
utilitare, funcții (Perl DBI), 674
utilizator rădăcină
conexiune cu serverul, 437
configurare, 437
parole (configurare), 437
utilizatori anonimi, 437
revocarea privilegiilor, 451

V

-V, opțiune de program, 600
valori (numerice), trunchiere, 109
valori ale anilor, ambiguitate, 132
valori ambigue ale anilor, interpretare, 132
valori caracter (de lungime variabilă), repartizare în coloane, 48
valori caracter de lungime variabilă, repartizare în coloane, 48
valori corecte (coloane SET), 125
valori cu semn (domeniile tipurilor de coloane numerice), 106-107
valori de date, 102
repartizare în coloane, 48
valori de date, identificare, 45
valori de tip oră, 102
valori fără semn (domenii ale tipurilor de coloane numerice), 106-107
valori incorecte (conversie de tip), 157
valori minime (întregi cu virgulă mobilă), 107
valori negative
întregi fără semn, 106-107
secvențe, 114
valori temporale, 102

Universitatea Tehnică Iași
BIBLIOTECA

valori trunchiate (dimensiunea coloanei), 109
VALUES(), listă (instrucțiunea SELECT), 577
VARCHAR(), tip de date, 103, 520
  combinare cu CHAR, 118-120
variabile. *Vezi și* variabile de mediu
  clientul mysql, 614
  instrucțiunea SHOW VARIABLES, 594
  interfața Perl DBI API
    convenții de denumire a variabilelor de manipulare, 295
    nume convenționale, 664
  PHP, 379
  serverul MySQL, configurare, 463
  serverul mysqld, 625-628
  utilitarele isamchk și myisamchk, 608-609
  utilitarul mysqldump, 631
variabile de manipulare (Perl DBI), 294-295
variabile de mediu, 594. *Vezi și* variabile
  Perl DBI, 679
variables, comandă (utilitarul mysqladmin), 620
variante de dezvoltare, 468
vederi, 194
--verbose, opțiune de program, 600
verificare
  privilegiile interogărilor, 480-481
  tabele la pornire, 442
verificarea erorilor, programe client pentru, 242-246
verificarea tabelelor, 490
  protocoale de blocare, 493
  scripturi, 496-497
VERSION(), funcție, 563
--version, opțiune de program, 601
versiuni
  myisamchk, 495
  MySQL
    actualizare, 467
    variante de dezvoltare, 468
  pachete, selectare, 502
--vertical, opțiune de program, 614

W-Z

-W, opțiune de program, 599, 607, 610, 614, 618
--wait, opțiune de program, 607, 610, 614, 618
WEEK(), funcție, 557
WEEKDAY(), funcție, 557
--where, opțiune de program, 630
WHERE, clauză
  instrucțiunea DELETE, 88, 571-572
  instrucțiunea SELECT, 64-65, 584
    echi-uniri, 178
  instrucțiunea UPDATE, 595
Windows
  aspecte privind licențele pentru MySQL, 16
  MySQL, instalare, 512
    Apache, 515
    distribuția client-server, 513-514
    MyODBC, 514
    PHP, 515
    suportul Perl DBI, 514-515
  nume de utilizator prestabilit, 39
WRITE, cuvânt cheie, 581
YEAR, tip de date, 104, 130, 524
YEAR(), funcție, 72-73, 557
ZEROFILL, atribut, 115
  coloane numerice, 110